MacTech Magazine
Vol. 20, No. 01 • January 2004

MacTech®

The Journal of Macintosh Technology and Development

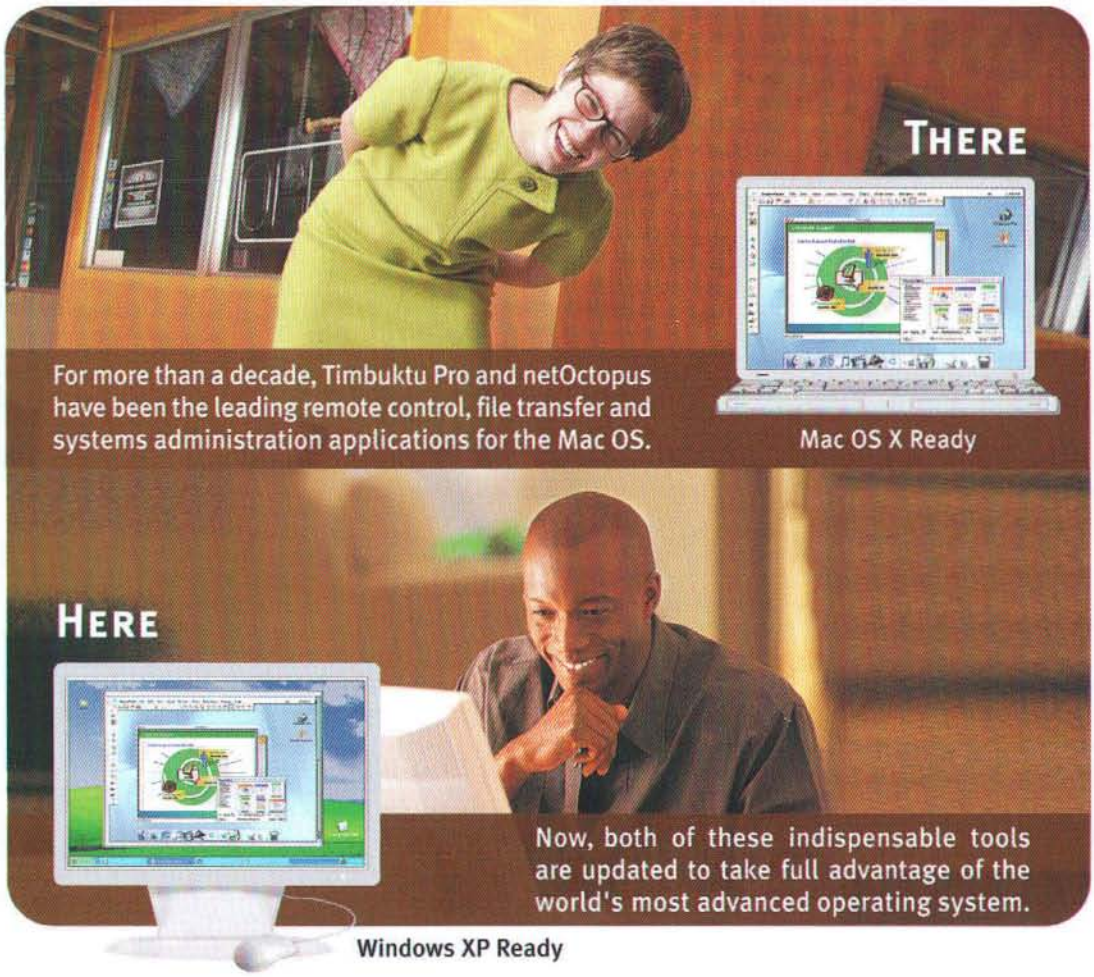
NICE KITTY.

A LOOK AT THE LATEST RELEASE OF MAC OS X

BY JOHN C. WELCH

$8.95 US
$12.95 Canada
ISSN 1067-8360
Printed in U.S.A.

Stay In Control Wherever You Go.



Timbuktu Pro

Whether you're at home or at work, Timbuktu Pro allows you to operate distant computers as if you were sitting in front of them, transfer files or folders quickly and easily, and communicate by instant message, text chat, or voice intercom.
http://www.timbuktupro.com

netOctopus

Intuitive and powerful, netOctopus can manage a network of ten or 10,000 computers. Inventory computers, software and devices on your network; distribute software; configure remote computers; and create custom reports on the fly.
http://www.netoctopus.com

Learn more, try it, or buy it online. Call us at 1-800-485-5741.

timbuktu® • netOctopus®

netopia.

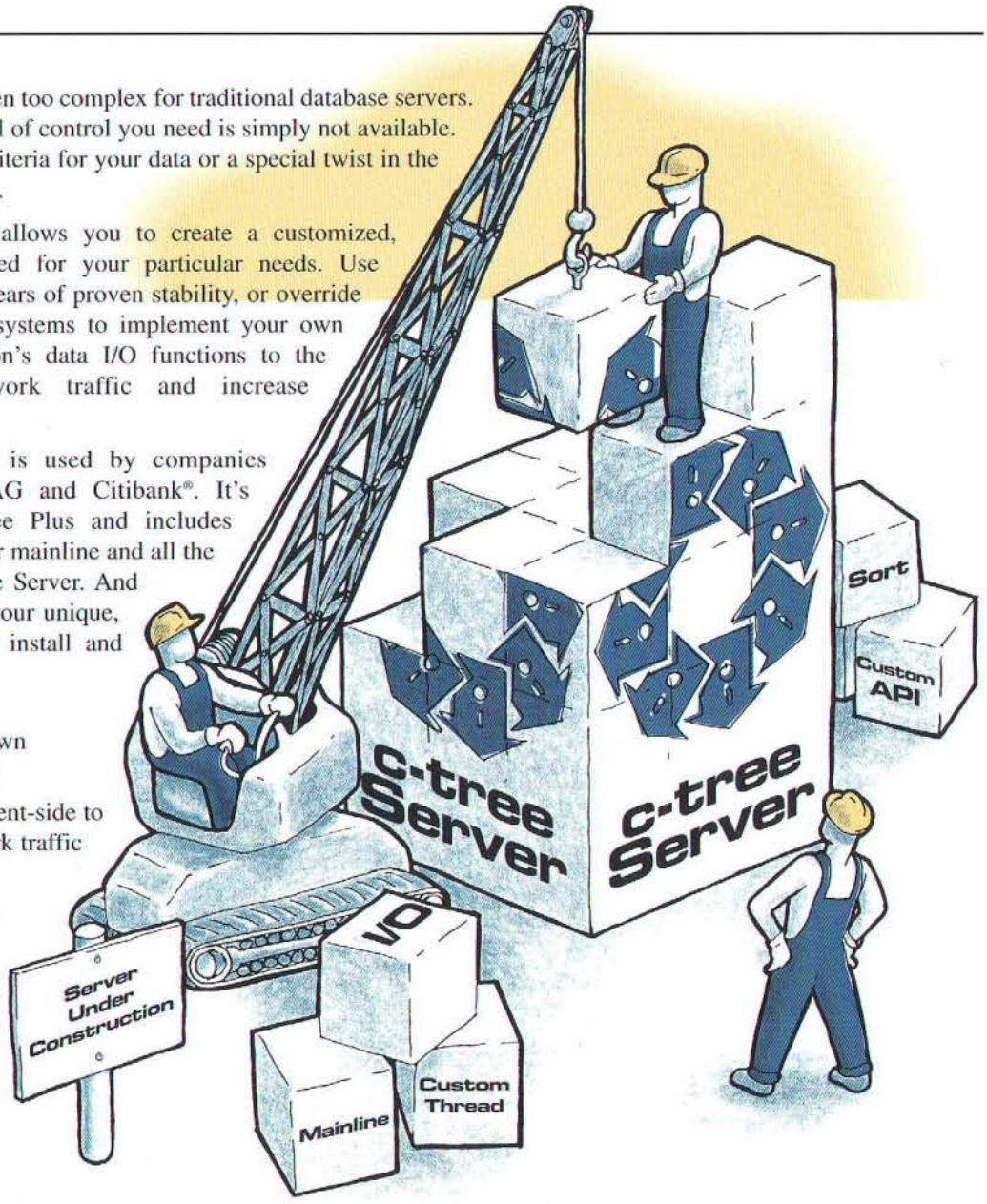
CUSTOMIZE
YOUR DATABASE SERVER WITH THE C-TREE® SERVER SDK

Today's database demands are often too complex for traditional database servers. The functionality and precise level of control you need is simply not available. Perhaps you need alternate sort criteria for your data or a special twist in the threading or communication logic.

FairCom's c-tree® Server SDK allows you to create a customized, industrial-strength server designed for your particular needs. Use FairCom's kernel, with over 20 years of proven stability, or override functionality within specific subsystems to implement your own subtleties. Move your application's data I/O functions to the server-side to decrease network traffic and increase performance!

FairCom's c-tree Server SDK is used by companies worldwide such as Software AG and Citibank®. It's integrated seamlessly into c-tree Plus and includes complete source code to the server mainline and all the interface subsystems to the c-tree Server. And best of all, once you've created your unique, customized server, it is easy to install and administer: no DBA required!

- Enhance our server with your own custom server-side functionality
- Move functionality from the client-side to the server-side to reduce network traffic and increase performance
- Modify or replace entire server subsystems
- Complete source for the server mainline, key server subsystems, and client-side
- Flexible OEM licensing



Visit www.faircom.com/ep/mt/sdk today to take control of your server!

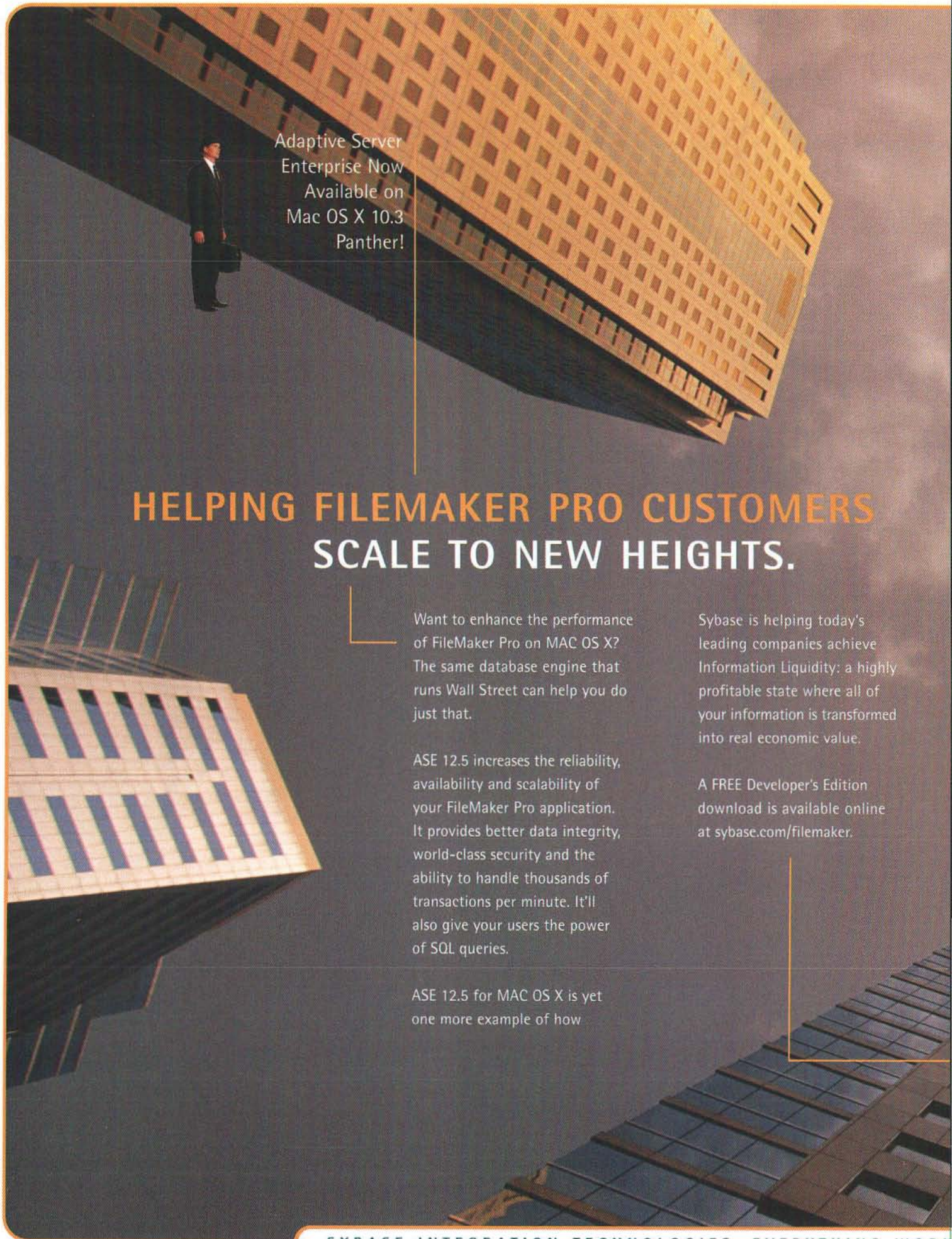
FairCom®
www.faircom.com

USA	573.445.6833
EUROPE	+39.035.773.464
JAPAN	+81.59.229.7504
BRAZIL	+55.11.3872.9802

DBMS Since 1979 • 800.234.8180 • info@faircom.com

Other company and product names are registered trademarks or trademarks of their respective owners.

© 2002 FairCom Corporation



HELPING FILEMAKER PRO CUSTOMERS SCALE TO NEW HEIGHTS.

Want to enhance the performance of FileMaker Pro on MAC OS X? The same database engine that runs Wall Street can help you do just that.

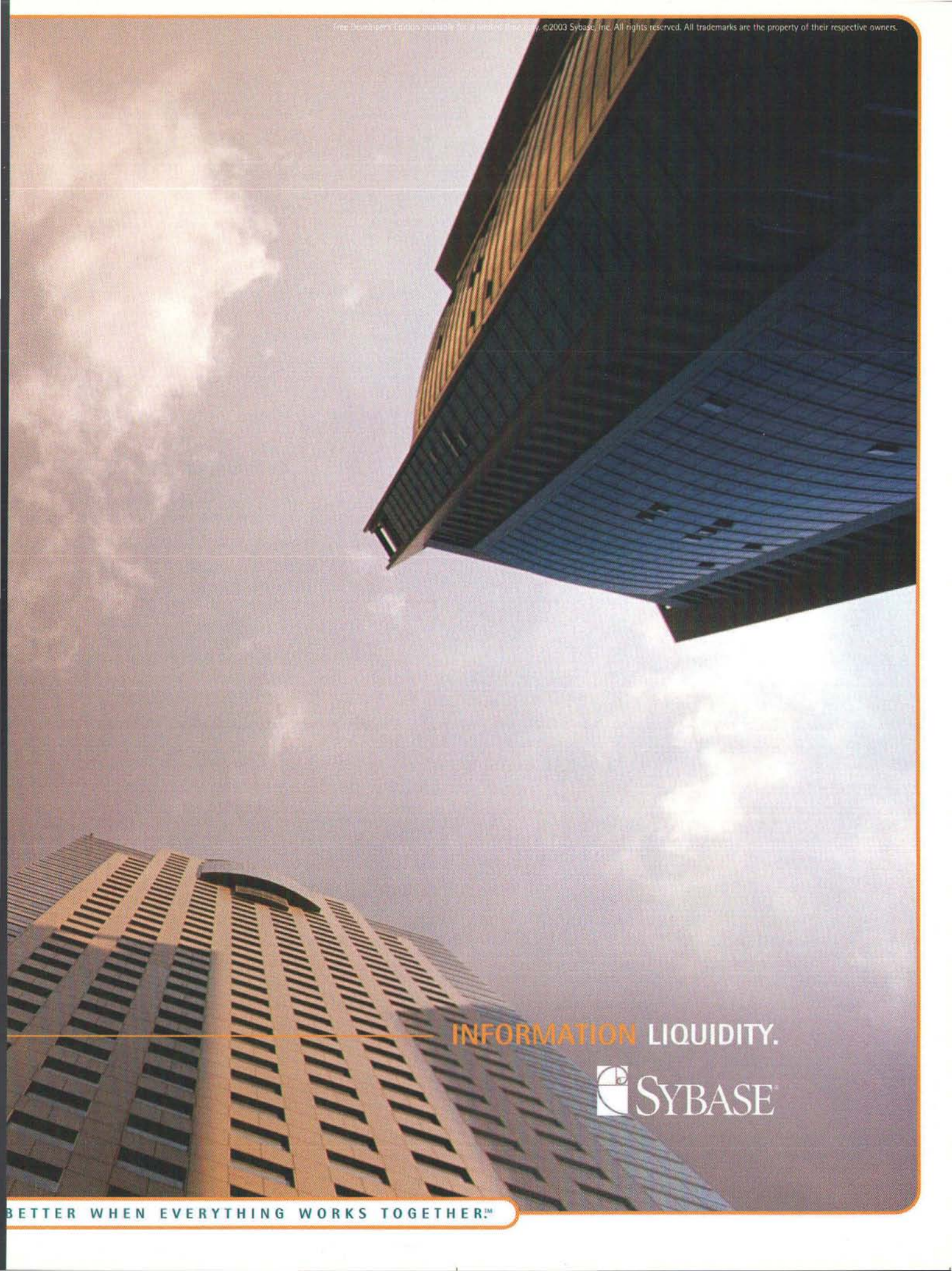
ASE 12.5 increases the reliability, availability and scalability of your FileMaker Pro application. It provides better data integrity, world-class security and the ability to handle thousands of transactions per minute. It'll also give your users the power of SQL queries.

ASE 12.5 for MAC OS X is yet one more example of how Sybase is helping today's leading companies achieve Information Liquidity: a highly profitable state where all of your information is transformed into real economic value.

A FREE Developer's Edition download is available online at sybase.com/filemaker.

SYBASE INTEGRATION TECHNOLOGIES. EVERYTHING WORK

Free Developer's Edition available for a limited time only. ©2003 Sybase, Inc. All rights reserved. All trademarks are the property of their respective owners.



MacTech®
The Journal of Macintosh Technology & Development

A publication of XPLAIN CORPORATION

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 877-MACTECH

DEPARTMENTS	E-Mail/URL
Orders, Circulation, & Customer Service	cust_service@mactech.com
Press Releases	press_releases@mactech.com
Ad Sales	ad_sales@mactech.com
Editorial	editorial@mactech.com
Programmer's Challenge	prog_challenge@mactech.com
Online Support	online@mactech.com
Accounting	accounting@mactech.com
Marketing	marketing@mactech.com
General	info@mactech.com
Web Site (articles, info, URLs and more...)	http://www.mactech.com

The MacTech Editorial Staff

Publisher • Neil Ticktin

Editor-in-Chief • Dave Mark

Managing Editor • Jessica Stubblefield

Regular Columnists

Getting Started
by Dave Mark

QuickTime ToolKit
by Tim Monroe

Mac OS X Programming Secrets
by Scott Knaster

Reviews/KoolTools
by Michael R. Harvey

Patch Panel
by John C. Welch

Section 7
by Rich Morin

Untangling the Web
by Kevin Hemenway

John and Pals' Puzzle Page
by John Vink

Book Reviews
by Ron Davis

Software Marketing
by Dave Wooldridge

MacTech's Contributing Editors

- Michael Brian Bentley
- Vicki Brown
- Marshall Clow
- John. C. Daub
- Bill Doerrfeld, Blueworld
- Andrew S. Downs
- Gordon Garb, Sun
- Ilene Hoffman
- Chris Kilbourn, Digital Forest
- Rich Morin
- John O'Fallon, Maxum Development
- Will Porter
- Avi Rappoport, Search Tools Consulting
- Chuck Shotton, BIAP Systems
- Cal Simone, Main Event Software
- Steve Sisak, Codewell Corporation
- Chuck Von Rospach, Plaidworks

MacTech's Board of Advisors

Jordan Dea-Mattson, Jim Straus and Jon Wiederspan

Xplain Corporation Staff

Chief Executive Officer • Neil Ticktin

President • Andrea J. Sniderman

Controller • Michael Friedman

Production Manager • Jessica Stubblefield

Production/Layout • Darryl Smart

Marketing Manager • Nick DeMello

Account Executive • Lorin Rivers
adsales@mactech.com • 800-5-MACDEV

Events Manager • Susan M. Worley

International • Rose Kemps

Network Administrator • David Breffitt

Accounting • Jan Webber, Marcie Moriarty

Customer Relations • Laura Lane, Susan Pomrantz

Shipping/Receiving • Joel Licardie

Board of Advisors • Steven Geller, Blake Park, and Alan Carsrud

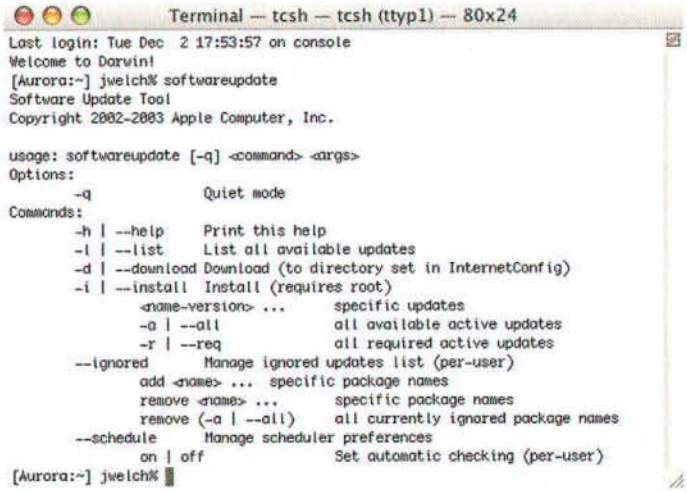
All contents are Copyright 1984-2004 by Xplain Corporation. All rights reserved. MacTech and Developer Depot are registered trademarks of Xplain Corporation. RadGad, Useful Gifts and Gadgets, Xplain, DevDepot, Depot, The Depot, Depot Store, Video Depot, Movie Depot, Palm Depot, Game Depot, Flashlight Depot, Explain It, MacDev-1, THINK Reference, NetProfessional, NetProLive, JavaTech, WebTech, BeTech, LinuxTech, MacTech Central and the MacTutorMan are trademarks or service marks of Xplain Corporation. Sprocket is a registered trademark of eSprocket Corporation. Other trademarks and copyrights appearing in this printing or software remain the property of their respective holders.

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

Contents

January 2004 • Volume 20, Issue 1



Page 38: *Command – Line softwareupdate in Panther*

Copyright 2003 by Dave Mark

Getting Started: Fun With AppleScript

A few weeks ago, I read an article talking about the changes to AppleScript introduced with the release of Panther. I've always been a big AppleScript fan, but this article really piqued my interest. I've been promising myself to spend some quality time digging into dictionaries, scripting environments, and especially using Interface Builder and AppleScript Studio to add Cocoa elements to AppleScript, and now that I've had enough quality play-time, I wanted to start writing about all this cool stuff.

Start by Setting Up Script Menu

Before we start playing with AppleScript itself, it is worthwhile taking a minute to install *Script Menu*, the little script icon that appears on the right side of the menu bar and gives you access to a wide range of AppleScripts.

Navigate over to *Applications/AppleScript/* and double-click on the script *Install Script Menu*. The Script Menu icon should appear in your menu bar. My Script Menu is shown in **Figure 1**. The Script Menu lists scripts from three different places on your hard drive. *Local Scripts* (also called *Library Scripts*) are found in the directory */Library/Scripts*. *User Scripts* are found in your home directory, inside */Users/<user name>/Library/Scripts/*. And, finally, *Application Scripts* are found within your *User Scripts* folder, in a subfolder called */Applications/<appname>/*, where *<appname>* is a folder with the exact same name as the application the scripts are written for.

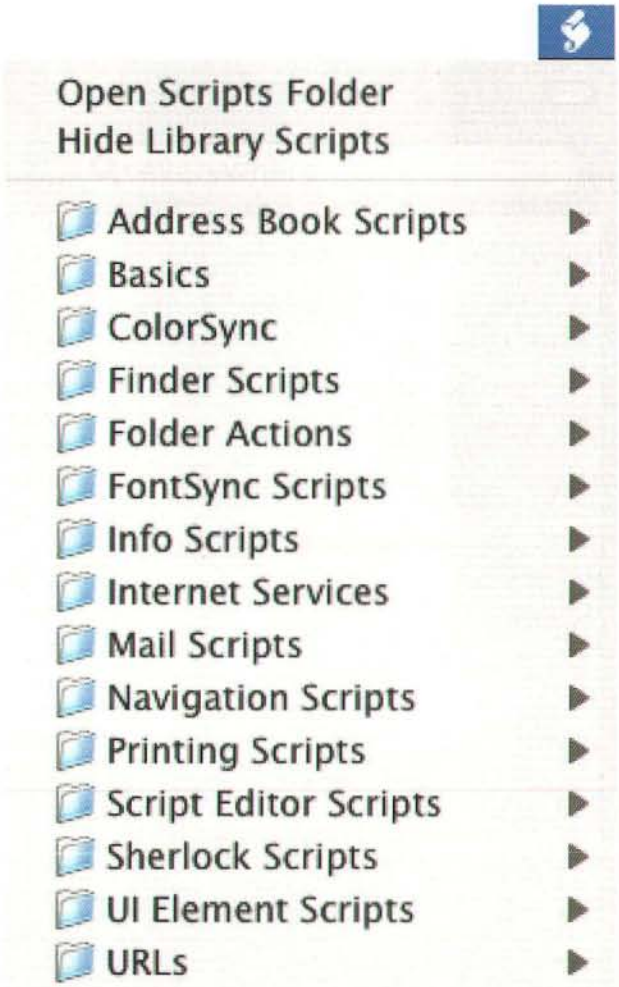


Figure 1. *The Script Menu's menu.*

MacTech MAGAZINE

Get MacTech delivered to your door at a price FAR BELOW the newsstand price. And, it's RISK FREE!

Subscribe Today!
www.mactech.com

Dave Mark is a long-time Mac developer and author and has written a number of books on Macintosh development, including ***Learn C on the Macintosh***, ***Learn C++ on the Macintosh***, and ***The Macintosh Programming Primer*** series. Be sure to check out Dave's web site at http://www.spiderworks.com.

Let's take a look at this in action. **Figure 1** shows the vanilla install of *Script Menu*. The first item, *Open Scripts Folder*, opens your *User Scripts* folder in a Finder window. Remember, *User Scripts* are the scripts you install in your home area.

The second menu item, *Hide Library Scripts*, removes the Library Scripts from the Script Menu.

Next comes a separator and the list of all the Library Scripts. In this case, all the scripts are divided into subfolders and so appear in the Script Menu divided into submenus.

Add in a User and Application Script

The screen shot in **Figure 1** shows Script Menu as it appears out of the box. Let's add a User Script to the menu. Go into the directory */Applications/AppleScript/* and launch the application *Script Editor*. Select *New* from the *File* menu, type in this simple script:

```
tell application "Finder"
  activate
end tell
```

Now select *Save* from the *File* menu and save the script in the directory */Users/xxx/Library/Scripts/*, where *xxx* is your user name. I saved my script under the name *Dave's Test Script*.

A quick note about terminology: The directory */Users/xxx/* is called your *home* directory and can be represented in the Unix world by the tilde character (~). For example, if I typed in the command:

```
ls /Users/davemark/
```

it would be exactly the same as typing:

```
ls ~
```

You can also use *~xxx* to refer to another user's home directory. So this is also equivalent way to refer to davemark's home directory:

```
ls ~davemark
```

And this is one way to list the contents of my User Scripts folder:

```
ls ~davemark/Library/Scripts
```

As soon as you save the script, it should appear in your Script Menu. Make sure you are still in Script Editor, then open the Script Menu, select your script (it should be at the very bottom), and watch what happens. Your script is executed, and it does what it is supposed to do. Namely, it tells the Finder to activate, to come to the front. And that is exactly what should happen.

New
PrimeBase 4.2
Replication Server

Check out the fully programmable
Replication Server

- Bidirectional Updates supported
- Update 3rd-party DBMS
- Send Emails
- Post/get Data to/from Websites

SQL-Runtime Plugin for REALbasic
$ 499,-

All PrimeBase Server Software

- SQL-Runtime Libraries available
- SQL Database Server
- Application Server
- Replication Server
- Open Server

Available on the most popular platforms

- Completely cross-platform
- Full-text searching and indexing
- Mac OS & OS X
- Linux
- Solaris
- IBM AIX
- all Windows platforms

PRIMEBASE

SNAP Innovation GmbH
Altonaer Poststraße 9a
D-22767 Hamburg / Germany
www.primebase.com
e-mail: info@primebase.com
Fon: ++49 (40) 389 044-0
Fax: ++49 (40) 389 044-44

Note that you could have created a folder in your *~/Library/Scripts/* folder and placed the script inside that folder. In that case, the folder would have appeared in the Script Menu as a submenu and the script would have appeared as an item inside that submenu. Try it!

Now for some ultimate coolness! Go back into Script Editor and close the window of the script you just created. Next, hold down the *option* key, click on the Script Menu and select your script again. Instead of running the script, Mac OS X opens the script in Script Editor. An excellent feature!

Your next step is to add an application-specific folder to your User Script directory. The idea here is to have a directory for all your scripted applications and have the scripts for that app appear in the Script Menu whenever that application is front most. You'll see what I mean in a minute.

Start off by creating a folder called *Applications* in your *~/Library/Scripts/* folder. Within that folder, create a folder with the exact name of one of your applications. For example, create a folder named *Finder*. Note that spelling is critical or this won't work. Now go into Script Editor and create this script:

```
tell application "Script Editor"
  activate
end tell
```

Save the script in the ~/Library/Scripts/Applications/*Finder* folder. I saved my script under the name *Dave's Finder Script*. Once the script is saved, click on the Script Menu. If you are in the Finder, your Finder scripts will appear at the bottom of the menu. If you are in another app, the Finder scripts will be replaced by the scripts (if any) for that app.

Figure 2 shows my Script Menu, as selected from within the Finder. Note that the User Script I created, *Dave's Script*, is second from the bottom, and the Finder script is at the very bottom. When I run *Dave's Script*, the Finder comes to the front. Then, within the Finder, when I run *Dave's Finder Script*, Script Editor comes to the front. Try this yourself.

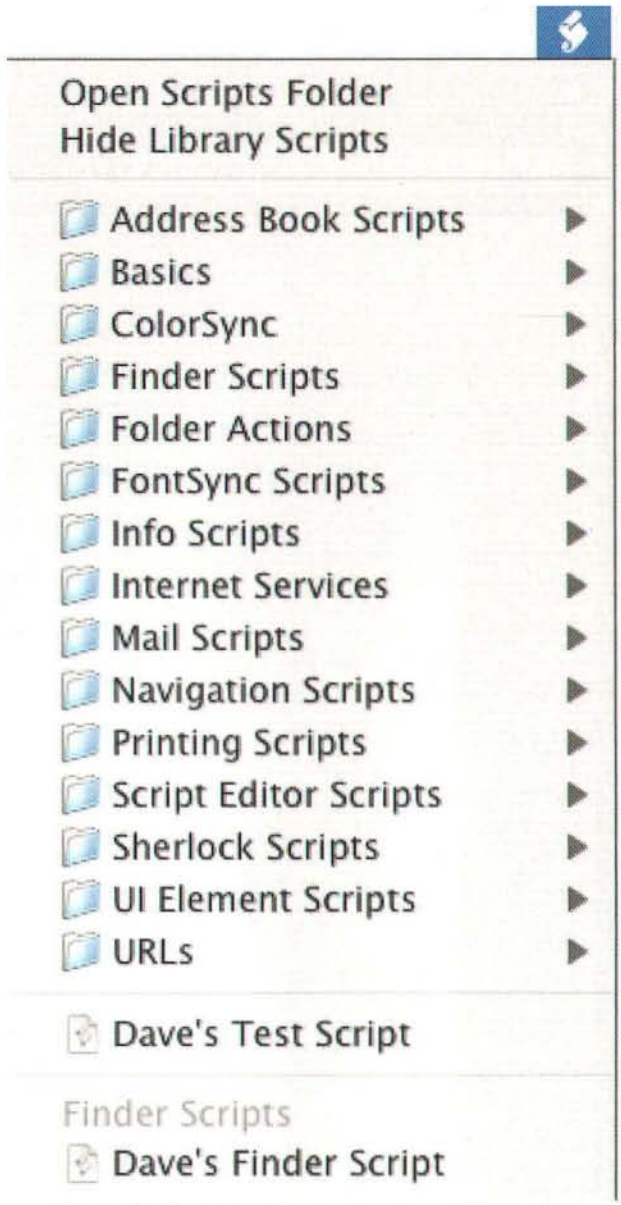


Figure 2. *The Script Menu with the addition of my test script from my User Scripts folder and my Finder Script from my Applications/Finder/ folder.*

GraphicConverter converts pictures to different formats. Also it contains many useful features for picture manipulation.

See **www.lemkesoft.com** for more information.

Apple's web site is full of cool sample scripts. This one will create a folder for the front most application in the *~/Library/Scripts/Applications/* folder:

http://www.apple.com/applescript/scriptmenu/downloads/addfrontapp.dmg

Exploring the Dictionary

Now that you have a home for all your scripts, let's start exploring AppleScript itself. One place to start is with an application's dictionary. Each application's dictionary entry gives

IN THE SOPHOS ZONE, VIRUSES NEVER INTERRUPT YOUR WORKFLOW

Sophos Anti-Virus provides total protection against all known viruses on Mac OS X. Its GUI enables users to perform an immediate scan of any accessible file, folder and volume, while InterCheck technology keeps users safe by intercepting all file accesses and scanning for viruses in the background in real-time. In addition, every license includes 24x7x365 technical support. In the Sophos Zone, viruses don't stand a chance.

For a free, fully supported evaluation, visit: www.sophos.com/av_mac

www.sophos.com/av_mac
Tel 888-216-6703

SOPHOS
engineered for business

specific information about the classes and commands that the application supports.

Launch Script Editor. If it is not already open, open the *Library* window by selecting *Library* from the *Window* menu. **Figure 3** shows my Script Editor library.

Figure 3. *Script Editor's Library window.*

Double-click on the *Finder* entry in the *Library* window. The window shown in **Figure 4** will appear. There is a *lot* of information to digest in this window. Spend a few minutes opening the various disclosure triangles (I've opened some of them in **Figure 4**) and clicking on the various commands and classes in the left scrolling pane.

If you've ever done any pre-OS X Mac development, you've likely encountered the concept of Apple events. Want an app to quit? Send it a *quit* Apple event. Want an app to open or print a file? Send it an *open* or *print* Apple event. AppleScript is basically a sophisticated language you can use to control a scriptable application by sending it events and scripting the results.

Figure 4. *The Finder's dictionary, showing the application class in the Finder Basics suite.*

The most basic set of commands are organized into something called the *Standard Suite*. Most applications support the *Standard Suite*. The *Standard Suite* commands are *close, count, data size, delete, duplicate, exists, make, move, open, print, quit*, and *select*.

Many applications add their own sets of commands to the Standard Suite. For example, the Finder adds in a set of classes and commands called *Finder Basics* and another called *Finder items*. The main pain of **Figure 4** shows the details of the *application* class in the *Finder Basics* suite. Notice that the list is divided into *Elements* and *Properties*. Elements are the objects contained in a class, while properties are more like preferences – unique within a class. Elements can be plural while a property is usually singular. A typical Finder element might be the list of items on the desktop. A typical Finder property might be the current selection.

Let's play with this a bit.

Create a new script and type in this code:

```
tell application "Finder"
  get home
end tell
```

When you run this script, Script Editor should display results in its *Result* pane. My results are shown in **Figure 5**. Notice that *home* is a Finder property containing the home directory. If you look towards the bottom of **Figure 4**, you'll see that it is also marked [r/o], meaning the property is read-only.

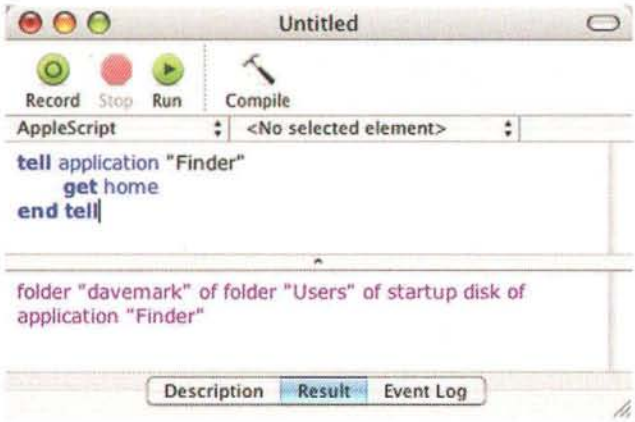


Figure 5. *A simple Finder script, with the results shown in the Result pane.*

Now edit the script like so:

```
tell application "Finder"
  get items of home
end tell
```

While the first script gave you the value of the *home* property, this script returns the list of items in the home folder. Here's the result I got when I ran this script:

```
{folder "Desktop" of folder "davemark" of folder "Users" of
startup disk of application "Finder", folder "Documents" of
folder "davemark" of folder "Users" of startup disk of
application "Finder", folder "Library" of folder "davemark"
of folder "Users" of startup disk of application "Finder",
folder "Magazines" of folder "davemark" of folder "Users" of
startup disk of application "Finder", folder "Movies" of
folder "davemark" of folder "Users" of startup disk of
application "Finder", folder "Music" of folder "davemark" of
folder "Users" of startup disk of application "Finder",
folder "Pictures" of folder "davemark" of folder "Users" of
startup disk of application "Finder", folder "Public" of
folder "davemark" of folder "Users" of startup disk of
application "Finder", folder "Sites" of folder "davemark" of
folder "Users" of startup disk of application "Finder"}
```

As you can see, this is a comma-delimited list, wrapped in curly braces. This kind of result is typical in AppleScript. As you can see in the top of **Figure 4**, an item element can be specified "*by numeric index, before/after another element, by name, as a range of elements, satisfying a test.*" For example, we can refer to item 3, as in this script:

```
tell application "Finder"
  get item 3 of home
end tell
```

As you might expect, here's the result of this script:

```
folder "Library" of folder "davemark" of folder "Users" of
startup disk of application "Finder"
```

Till Next Month...

There is a *ton* of cool stuff to play with here. In the Finder dictionary, look in the *Finder items* suite, at the *item* class to learn everything you could possibly want to know about items. For example, you'll see a *bounds* property, which describes the bounding rectangle of the specified item. You might extend the script above to:

```
tell application "Finder"
  get bounds of item 3 of home
end tell
```

You get the idea. The dictionary is an incredibly powerful tool for unlocking the mysteries of AppleScript. Play more. I'll be back next month with more fun stuff! ☺

By Scott Knaster

Ship It! Distributing Your Software

Here's a twist: let's assume your world-changing Mac OS X application is finished, documented, freshly painted, and ready to go out into the world. What are you going to do now? Why, you need to distribute your software, of course. We're going to take a couple of columns to look at how you get your software out into the world and installed on happy Macs everywhere.

There are two ways you can distribute your software: as a disk image, or wrapped in an installer. Mac OS X comes with Apple-supplied tools you can use to create both kinds of distributions, whether your software is going to get to users on a CD or via the Internet. In this month's column, we're going to cover the first kind of distribution: disk images.

In Your Image

If your software has just an application bundle to install, maybe along with some documentation or a couple of other ancillary files, you should consider providing drag-and-drop installation with a disk image. A disk image is simply a file that contains an encoded representation of a disk volume. Mac OS X can turn a disk image into something that looks exactly like a mounted volume. When you distribute a disk image online with your application on it, your users can download the disk image, have OS X mount it like a disk, then just drag your application from the volume and drop it on their own disks. You can also use a disk image to create CDs to distribute. When users get the CDs, they can drag and drop to install in the same way.

Writing and testing your software is hard – making a disk image that includes your software is very easy. When you're ready to start messing around with disk images, launch Disk Utility, which you'll find in the /Applications/Utilities/ folder. (If you're using Mac OS 10.2, use Disk Copy instead – the disk image creation features are in there. They're very similar to those in Disk Utility, but not identical. As a MacTech reader, we trust you can adapt.)

Note that even though Apple replaced the features of Disk Copy with Disk Utility in 10.3, not all of Disk Copy's features made it in. In particular, Disk Copy provided more options for your disk image format, and Disk Copy let you create an image from a folder by dragging the folder in (Disk Utility lets you do this via a menu item). Maybe these features will be revived in future versions of Disk Utility.

To make your new disk image, click New Image in the toolbar. You'll see a sheet like the one in **Figure 1**. Give your disk image a pleasant name. For the Size popup, choose a size that's big enough to hold the software you're going to put on it. It's OK if the virtual disk is way too big – if you're going to distribute it via the Internet, you can compress it later, so unused space won't be wasted. Leave the Encryption and Format settings as they are, and click Create. Disk Utility will show you a little progress dance, and then voila! You've created your first disk image (see **Figure 2**). After the image is created, OS X mounts it as a volume.

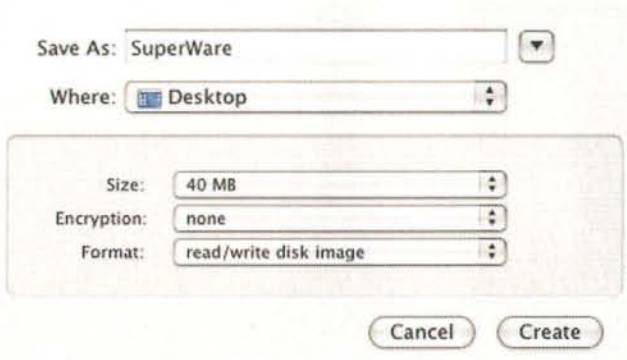


Figure 1. *Disk Utility offers various settings when you create your disk image. Usually, you'll just have to worry about the size, making sure it's big enough for your files.*

Scott Knaster has been writing about Macs for as long as there have been Macs. Scott wrote developer books for General Magic, worked on SDK stuff for Danger, and contributed to the first user manual for the Philips Velo. Scott's hobby is gaining and losing weight.

CHOSEN Nº.1

FOR SOFTWARE PROTECTION
BY SOFTWARE DEVELOPERS
WORLDWIDE.

HASP®
PROFESSIONAL SOFTWARE PROTECTION

Because of its unparalleled security, unsurpassed reliability, and ability to increase your software revenues, HASP4 is considered the world's best hardware-based software protection. No other company is more committed to the continual advancement of this proven technology for your success. Call 1-800-562-2543 or visit HASP.com for your FREE HASP4 Developer's Kit today.

North America: 1-800-562-2543, 847-818-3800 or HASP.us@eAladdin.com **International:** +972-3-636-2222 or HASP.il@eAladdin.com
Germany: HASP.de@eAladdin.com **UK:** HASP.uk@eAladdin.com **France:** HASP.fr@eAladdin.com **Benelux:** HASP.nl@eAladdin.com **Japan:** info@Aladdin.co.jp

©2003 Aladdin Knowledge Systems, Ltd. HASP is a registered trademark of Aladdin Knowledge Systems, Ltd.

Aladdin
SECURING THE GLOBAL VILLAGE
eAladdin.com

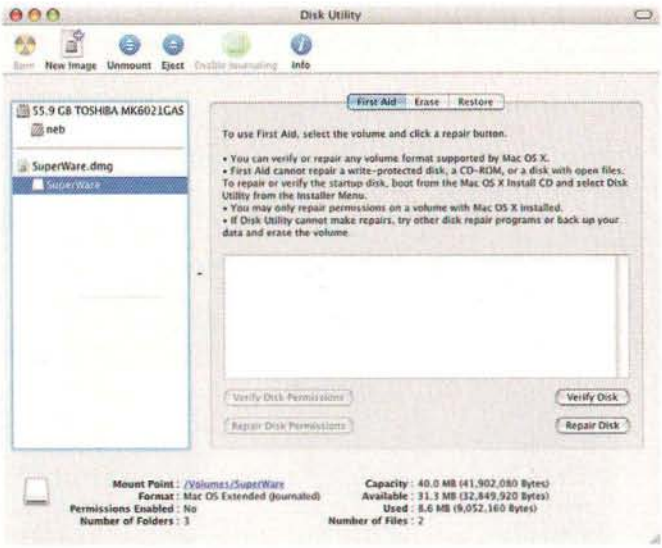


Figure 2. *After you create your disk image, it appears in the left pane of Disk Utility and its image is mounted.*

Build Your Case

Once you've created the disk image, switch to the Finder. You'll discover the Finder already knows about the volume supplied by your fresh new disk image. If you're running 10.3 and you have the Sidebar set to show removable media, your volume appears in the list. Now you need to add your files to it. Grab the files you want to distribute and drag them to the new disk.

After you've dragged and dropped the files, you can channel your inner designer and make the Finder window look just the way you want. Move the icons around to form a lovely pattern. Choose View --> Show View Options and add a background image or color, pick a different size for the icons, and make the labels look perfect.

Figure 3. *In the Finder, drag the files you want to the disk image, then go wild dressing up the contents however you like.*

Here are a couple of tips about prettying up the way your Finder window looks. The General panel of Finder preferences has a setting called *Open new windows in column view*. If the user has this checked, your disk image's window will indeed open in column view, which conceals all your painstaking icon placement and other visual tweaks. Sigh. Second, if you want to use a background picture (JPG, PNG, and so on) for your Finder window, make sure you copy the picture's file to your disk first, then use the Select button in the View Options window to point at the new copy. If you don't copy it to your disk first, it won't be there for users to see when they download your disk image or open your CD.

Once you have the files and the view settings you want, eject the disk image in the Finder and go back to Disk Utility. If you're going to distribute your software by CD, you can go ahead and order the pizza, because you're almost done. All that's left now is to burn the CD. You can do that from Disk Utility. Just select the disk image in the left pane and click Burn on the toolbar. Disk Utility will ask you to insert a blank CD, then give you the dialog you see in **Figure 4**. Just click Burn and wait for your CD to be made.

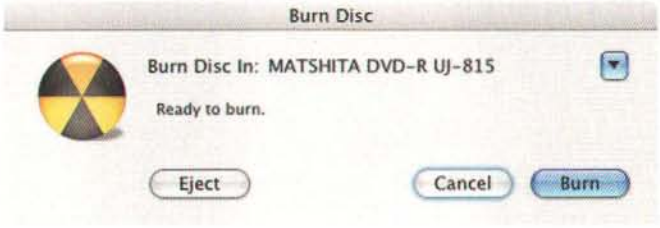


Figure 4. *"Ready to burn," indeed! You'll get this dialog when you have selected a disk image (.dmg) and inserted a blank CD.*

Download Lowdown

If you want to get your disk image ready for online distribution instead of via CD, the first thing to do is get that extra space out of the disk image. To do this, while still in Disk Utility, select the disk image in the left pane (if it's not there, you can drag it in from the Finder, or choose Images --> Open), and then choose Images --> Convert. Pick Compressed as the image format, and let 'er rip. Soon you'll have your compressed image. (For fun, you can go into the Finder and Get Info on both the original and compressed images. You'll see that the original is 40 Mb, or whatever other value you specified when you created it, and the compressed version not only gets rid of the unused space, it compresses the files, so the image uses less space than the original files.)

You might notice that Disk Utility keeps track of all the disk images it has ever seen on its little shelf over in the left pane. The way to get rid of those isn't obvious: just drag them off and drop them anywhere but the pane, and they'll go poof.

Once you have the compressed image, you're done. You can post it online for downloading. Let's take a look at the

experience users have when downloading a disk image with a web browser. If you download a disk image with a non-Apple browser, such as Internet Explorer or Camino, here's the typical experience:

1. User clicks a link to download the disk image file. The file goes into the Downloads folder.
2. In the Finder, in the Downloads folder, the user double-clicks the disk image. The Finder decompresses the disk image and mounts it. Usually, the disk automatically opens.
3. The user drags the downloaded application to the hard disk, usually to the Applications folder.
4. The user ejects the mounted disk image.
5. The confused, exhausted user drags the disk image file to the trash.

When Safari appeared in January 2003, Apple made it smarter than the average bear for downloading disk images. Safari knows how to decompress and mount disk images all by itself, without having to ask the Finder for help. So when you download a disk image with Safari, there's one less step: you don't have to switch to the Finder and double-click the disk image file. That's progress.

Becoming an Enabler

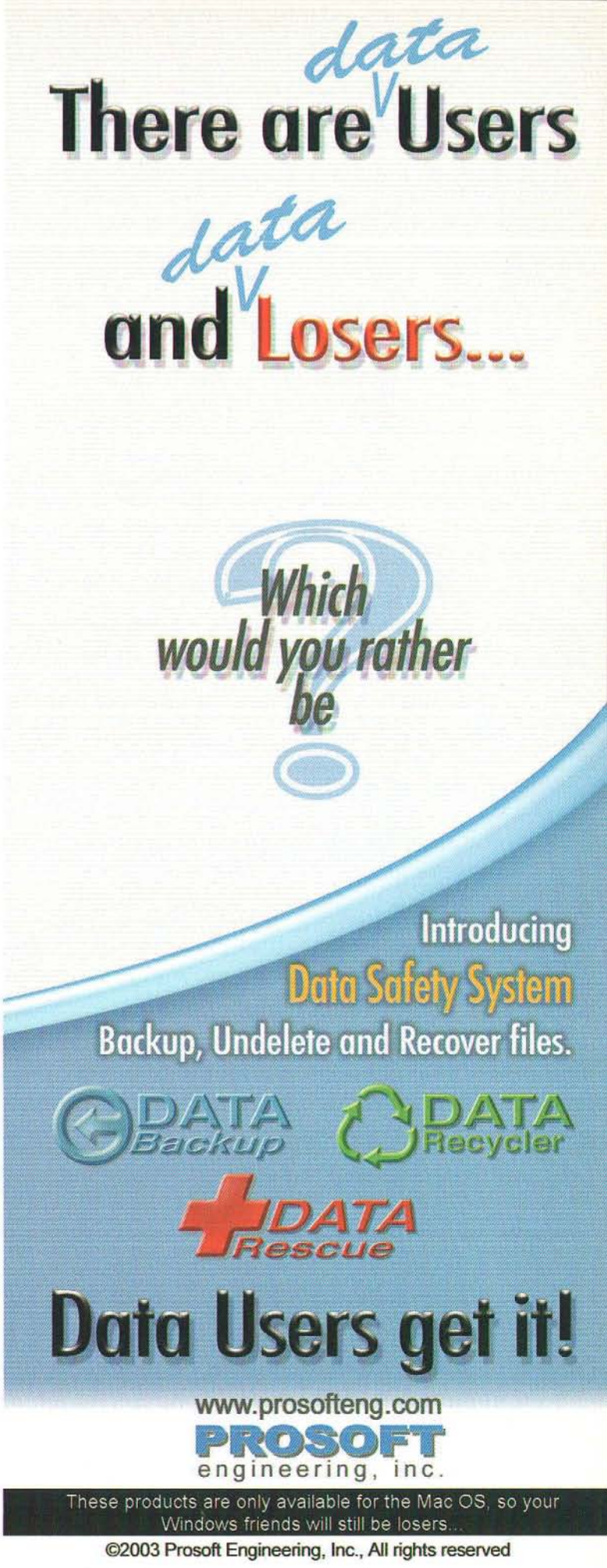
When Mac OS 10.2.3 arrived, Apple made the download experience even better, as long as you were using Safari. Mac OS 10.2.3 introduced the groovy concept of **Internet-enabled disk images**, which make the download experience even better. When users download an Internet-enabled disk image, Safari decompresses the file and mounts the virtual disk, as usual. But then, it continues by copying the disk's contents to the Downloads folder, ejecting the mounted disk image, and moving the disk image file to the trash. That's pretty cool for users. All they have to do at that point is visit the files in the Downloads folder and copy them to their final destination. And Internet-enabled disk images are compatible with all browsers and Mac OS X versions. Users who are not running Safari with Mac OS X 10.2.3 or later will still get the old, standard behavior.

How do you make an Internet-enabled disk image? I was hoping you would ask that question. It's free and oh-so-easy. You merely have to issue a single instruction using hdiutil, a command line utility that comes with Mac OS X. To Internet-enable your favorite disk image, just go to the Terminal and type

```
hdiutil internet-enable -yes SuperWare.dmg
```

Of course, make sure you have the right path to your disk image file. As always, you can drag the file into the Terminal window from the Finder if you want to be sure. If you do it right, hdiutil will confirm back to you thusly:

```
hdiutil: internet-enable: enable succeeded
```

That's all there is to it. Sure, it would be nice if Disk Utility had a command for Internet-enabling, and it surely will in the future, but for now, this is how we do it. If you find distasteful even this extremely brief journey into Unix, you can use the donationware program IDIot, which makes it super-simple to Internet-enable a disk image. IDIot is available at http://www.ifthensoft.com/idiot.hqx . And if you want to read all about Internet-enabled disk images from the folks who invented them, see http://developer.apple.com/ue/files/iedi.html.

Master and Commander

As we have seen, Disk Utility makes it pretty darn easy to create disk images, and with just a little tweak under the hdiutil hood, we can improve the user experience by making disk images Internet-enabled. If you're the command-line type, or you need to write shell scripts to manipulate your disk images, you should consider hdiutil for more than just the bit part described above.

You can perform the same basic functions with hdiutil as you can with Disk Utility, plus lots more geeky stuff. For example, earlier we used Disk Utility to create a disk image, copy files to the image, then compress it for optimum online distribution. Let's do the same steps with hdiutil. First, we create the disk image itself and ask the operating system to mount it:

```
[neb:~] scott% hdiutil create -size 40m -fs HFS+ -volname
"SuperWare" SuperWare.dmg

Initializing...
Creating...
.....................................................................
.................
Formatting...
Finishing...
created: /Users/scott/SuperWare.dmg

[neb:~] scott% hdiutil mount SuperWare.dmg
Initializing...
Attaching...
Finishing...
Finishing...
/dev/disk2              Apple_partition_scheme
/dev/disk2s1            Apple_partition_map
/dev/disk2s2            Apple_HFS
/Volumes/SuperWare
```

Note the aid and coddling that Disk Utility gave us when we used it previously. Disk Utility thoughtfully assumed we needed an HFS+ disk, wanted the volume name to be the same as the filename of the image, and would like to mount the new volume. When we use hdiutil, we have to explicitly choose the file system and volume name we want, and we need a second command to mount the volume. Of course, this little bit of extra work also indicates that hdiutil provides us with extra flexibility. For example, you can choose a different file system, or a volume name that's not the same as the file name when you use hdiutil. As an extreme example of the flexibility of hdiutil, it lets you specify the size of the disk image in various units, including megabytes, gigabytes, terabytes, petabytes, and exobytes. An exobyte is a gigabyte of gigabytes. That's thinking ahead!

At this point, we're ready to copy the desired files to the disk image. Of course, we can use the command line or the Finder for this, or both. You'll probably want to use the Finder for delicate icon positioning and tweaking Finder view settings. Once the disk image has everything you want, unmount it:

```
[neb:~] scott% hdiutil unmount /Volumes/SuperWare
hdiutil: unmount: LetIOKitSettleDown: (timed out)
"disk2s2" unmounted successfully.
```

After unmounting, we can use hdiutil to burn a CD, like so:

```
[neb:~] scott% hdiutil burn SuperWare.dmg

Preparing data for burn
Opening session
Opening track
Writing track
.....................................................................
.................................
Closing track
Closing session
Finishing burn
Verifying burn...
Verifying
.....................................................................
.................................
Burn completed successfully
.....................................................................
.................................

hdiutil: burn: completed
```

Burning a CD with hdiutil is like a game: you see how many dots you can get before more text appears. No fancy-pants Aqua progress bars here! If you're not going to burn a CD, but you instead want to prepare the disk image for downloading, you need to compress it, as we did with Disk Utility. With hdiutil, that works as follows:

```
[neb:~] scott% hdiutil convert -format UDCO -o
SuperWare_online.dmg SuperWare.dmg

Preparing imaging engine...
Reading DDM...
   (CRC32 $767AD93D: DDM)
Reading Apple_partition_map (0)...
   (CRC32 $DD66DE0F: Apple_partition_map (0))
Reading Apple_HFS (1)...
.....................................................................
.................................
   (CRC32 $9F4C65C5: Apple_HFS (1))
Reading Apple_Free (2)...
.....................................................................
.................................
   (CRC32 $00000000: Apple_Free (2))
Terminating imaging engine...
Adding resources...
.....................................................................
.................................
Elapsed Time:  1.074s
 (1 task, weight 100)
File size: 31489 bytes, Checksum: CRC32 $A3B0383C
Sectors processed: 81920, 1345 compressed
Speed: 626.0Kbytes/sec
Savings: 97.6%
created: SuperWare_online.dmg
```

In amongst all that geeky progress reporting, we've created our compressed image in the file SuperWare_online.dmg. The **–format UDCO** option is the way you create a compressed image. And by the way, if you would rather not see Terminal spew all that stuff at you, most **hdiutil** commands include a **–quiet** option to suppress output. To find out about all the features of **hdiutil** – and there are an enormous number of them – check out man **hdiutil**. Most **hdiutil** commands have their own individual help commands, which you can get by typing **hdiutil**, the command name, and the **–help** option, like this:

```
[neb:~] scott% hdiutil unmount -help
hdiutil unmount: unmount a mounted partition
Usage:  hdiutil unmount <mountpoint>
        Options:
            -force                  force unmount

        Common options:
            -verbose
            -debug
            -quiet
```

Another powerful option for creating disk images is DropDMG, a $10 utility written by Michael Tsai. This program provides a nice Aqua UI with more disk image creation features than Disk Utility, has a thorough AppleScript dictionary, and is also available from the command line. You can get DropDMG at http://www.c-command.com/dropdmg/index.shtml.

You might also be interested in BootCD, a free utility for making disk images that create bootable CDs. BootCD is available at http://www.charlessoft.com/.

Coming Soon

As we mentioned at the start, disk images are only one way of passing out software. Disk images are great for distributions that are simple enough to allow drag-and-drop installation. However, drag installing leaves behind older copies of the application, which can confuse the Finder. And if you have more complex requirements, such as needing to put files in various places on the disk, run code while you're installing, or install on OS 9 as well as OS X, you'll need an installer.

You can use the PackageMaker and Installer tools provided by Apple, or get a more powerful third-party tool, such as Installer VISE, VISE X, or FileStorm from MindVision, or StuffIt InstallerMaker from Aladdin. Many commercial developers prefer those third-party programs for complex installations, but Apple's built-in tools are easy to use, improving, and free – if you have OS X, you already own them. In next month's column, we'll take a look at how to use these tools to create your own installers. Until then, get your software coded, tested, debugged, and documented, so you'll be all ready to distribute it.

By Kevin Hemenway, Oscillating Undulation

Your First MySQL Database

Creating, administering, and maintaining are pretty easy.

Last month, besides exploring what was different between the Jaguar and Panther web serving configurations, we installed the MySQL database program (specifically, the "value-added" installer from third-party Server Logistics), used the provided System Preference to initialize the database and set the MySQL root password, then confirmed it was running through one of three avenues (log files, the Terminal, or the Activity Monitor).

We've yet to create a MySQL database or insert any data. We betta' rECtififfYYyy!

Exploring The MySQL Helper Programs

Along with the MySQL daemon (where "daemon" is an application that runs all the time, resolutely waiting for something to do), a bunch of other useful helper programs were installed alongside your new database server. Take a look in your /Library/MySQL/bin directory (**Figure 1**) for the complete list. Some of these programs (like msql2_mysql, make_win_src_distribution, mysql_install, and mysql_install_db) will never be used in normal (or even abnormal) operation. Others (like mysqladmin, mysql_setpermission, mysqlshow and mysqldump) will become regular additions to your MySQL administrating repertoire.

The most negative aspect of these helper utilities is that there's no central place to find information on what they all do. Some, like mysql_config, have no explanations for their purpose (commonly accessible by passing -?, -h, or --help as command line flags), but their source code can be viewed with vi, less, or BBEdit to divine their intent. Others, like mysqlshow, have manual pages that can be accessed with a command like man –M /Library/MySQL/man/ mysqlshow (see "Homework Malignments" for a shortcut), and still others, like mysql_setpermission, are Perl scripts that need an additional module (DBI) not installed by default.

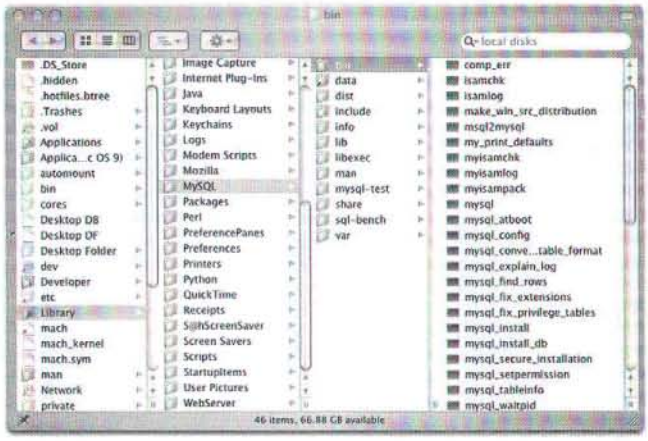


***Figure 1:** A number of additional MySQL utilities were installed.*

Since the DBI Perl module plays an important part (both in the mysql_setpermission script and any future database interaction in Perl), we'll install that onto our machines before we go much further. Prior to starting, however, we'll need to fix a bug that wasn't corrected in time for Panther's public release. Open /System/Library/Perl/5.8.1/darwin-thread-multi-2level/Config.pm in an authenticating text editor (I prefer BBEdit, but you can use something like sudo vi $filename), and look for the following line:

```
ld='MACOSX_DEPLOYMENT_TARGET=10.3 cc'
```

Add the word env, creating the following correction:

```
ld='env MACOSX_DEPLOYMENT_TARGET=10.3 cc'
```

Kevin Hemenway, coauthor of *Mac OS X Hacks* and *Spidering Hacks*, is better known as Morbus Iff, the creator of disobey.com, which bills itself as "content for the discontented." Publisher and developer of more home cooking than you could ever imagine (like the popular open-sourced aggregator AmphetaDesk, the best-kept gaming secret Gamegrene.com, the ever ignorable Nonsense Network, etc.), he has started to self-teach indexing and cataloging skills for his . .. . . "media collection". Contact him at morbus@disobey.com.

After saving the file, we've got one more annoyance to take care of. Remember how we set the MySQL root password so that our database would be reasonably secure? Welp, the Perl module we're about to install requires access to MySQL to run some tests, and there's no simple way to give it what it needs without temporarily blanking out that root password. Using the installed MySQL System Preference doesn't allow us to set a null value, so we'll need to run the following in the shell: **/Library/MySQL/bin/mysqladmin -u root -p password ""** (no spaces between those quotes). You'll be prompted for your current password... once entered, the root password will be set to nothing.

We're now ready to start the DBI installation process. To do so, we're going to use Perl's CPAN (the "Comprehensive Perl Archive Network") to automatically download the module, check and enable any prerequisites, ensure everything is working properly with a bevy of tests, and finally, install the modules for regular use.

Normally, to start CPAN, we'd simply enter **sudo perl –MCPAN –eshell** at the command line. However, the module we'll be installing needs to know a bit about our MySQL installation so, just this once, we're going to use the following commands instead. These will, for the duration of our current shell, add the MySQL **bin** directory to the lookup path (instructions on how to set these permanently are available in "Homework Malignments"):

```
# if you're using the tcsh shell:
setenv PATH ${PATH}:/Library/MySQL/bin
sudo perl -MCPAN -eshell

# if you're using the bash shell, use the following:
PATH=$PATH:/Library/MySQL/bin sudo perl -MCPAN -eshell
```

If this is your first time using CPAN, you may be asked oodles and oodles of configuration questions, including whether you're using proxies, what CPAN sites you want to download from, and so on. In most cases, you can just accept the defaults. Eventually, you'll end up at a command prompt that looks something like this:

```
shell -- CPAN exploration and modules installation (v1.76)
ReadLine support enabled

cpan>
```

Now, type **install Bundle::DBD::mysql**. This will install a bunch of modules related to the MySQL DBD ("database definition") of Perl's DBI ("database interface")—you'll see a dozen screens of information fly by before the process is finished. Depending on your CPAN configuration, you may be asked to follow some missing module prerequisite, which you should generally always agree to.

If the CPAN process complains about its inability to download modules from any of the mirror sites you chose during configuration, you probably have Panther's internal firewall enabled. Cancel the CPAN process, execute **export FTP_PASSIVE=1** in the shell, and then start the CPAN process again. More information is available in "Homework Malignments".

If any critical errors occur, the install process will stop... in our case, we should have received a few during the testing phase of DBD::mysql (**Figure 2**). We could go nuts about actually hunting down and fixing these errors that caused the integrity checks to fail (thus canceling the installation), but honestly, you can "cheat" and force things forward anyways. Some will cluck at me for saying so, but run the following CPAN command to force the installation: **force install Bundle::DBD::mysql**. For more information about the failing tests, reference http://www.mail-archive.com/macosx@perl.org/msg05834.html.

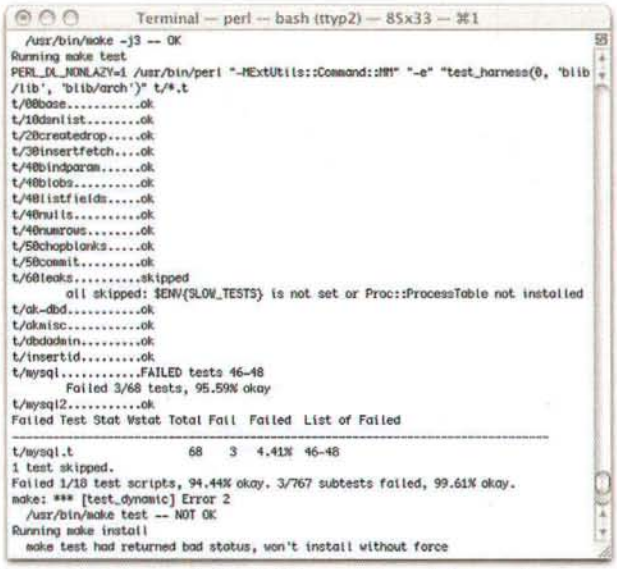


Figure 2: *Failures in the DBD::mysql module can be ignored.*

Once the forced installation has finished, you'll be returned to the standard cpan> prompt. Type exit to finish the process, and then be sure to reset the MySQL root password back to what it was before. You can do this either from the command line (using /Library/MySQL/bin/mysqladmin -u root password "*password*") or by using the MySQL System Preference (**Figure 3**).

Figure 3: *Using the System Preference to set the MySQL root password.*

Creating Our First Database

The larger purpose of getting the Perl DBI and DBD modules installed was to use one of the MySQL helper utilities to create a new database, a new MySQL user, and set the proper permissions for access thereafter. There are many different ways and avenues this can be done; I happen to think that mysql_setpermission, a utility that requires DBI, is one of the easiest and friendlier paths to speed down.

We can start the mysql_setpermission script one of two different ways. Without any additional command line flags, it will attempt to connect to the MySQL database as the current user: morbus, in my case. Since that user has yet to be created within MySQL (remember, MySQL users and permissions have nothing to do with Linux users and permissions), I'll receive a disheartening error about lack of access:

```
~ > /Library/MySQL/bin/mysql_setpermission
Password for user to connect to MySQL:
Can't make a connection to the mysql server.
 The error: Access denied for user: 'morbus@localhost'
 (Using password: YES) at /Library/MySQL/bin/mysql_
 setpermission line 70, <STDIN> line 1.
```

We can solve this one of two ways: temporarily becoming the root user with sudo /Library/MySQL/bin/mysql_setpermission, or by passing the MySQL username as a command line flag with /Library/MySQL/bin/mysql_setpermission --user root. Regardless of the method, we'll reach a menu with numerous possibilities (**Figure 4**).

Figure 4: *The starting menu of* mysql_setpermission.

Since we've yet to create a database or MySQL user, we'll want to choose the second option, which allows us to do both. You'll be asked the name of the database to create, the username and password for the new MySQL user (passwords are heartily recommended if you plan on allowing the new user to modify data), and the hosts this user can access the database from. In our example (**Figure 5**), the new mactech database can be accessed by davemarksman from any host (represented by the % character). The host restriction determines whether *other programs on other servers* can connect to the mactech database. When a normal web visitor accesses any of our connecting code, it's considered *localhost access* (the visitor uses our

Multiple formats. Multiple platforms. Complex installers.

Aladdin solves the compression and installation puzzle.

Trying to figure out how to handle multiple compression formats and platforms?
The StuffIt Engine solves the compression puzzle.

Aladdin's StuffIt Engine SDK:

- Adds value to your application by integrating powerful compression and encryption.
- Is the only tool that supports the StuffIt file format.
- Provides a single API that supports over 20 compression and encoding formats common on Macintosh, Windows, and Unix.
- Makes self-extracting archives for either Macintosh or Windows.
- Available for Macintosh, Windows, Linux, or Solaris.

Licenses start as low as $99/year

To learn more, visit: www.stuffit.com/sdk/

StuffIt Engine SDK™

The power of StuffIt in your software.

Looking for the easiest and fastest way to build an installer?
StuffIt InstallerMaker completes your puzzle.

It's not enough just to write solid code anymore. You still have to write an installer for your users. StuffIt InstallerMaker makes it simple and effective.

- StuffIt InstallerMaker gives you all the tools you need to install, uninstall, resource-compress or update your software in one complete, easy-to-use package.
- Add marketing muscle to your installers by customizing your electronic registration form to include surveys and special offers.
- Make demoware in minutes. Create Macintosh OS X and Macintosh Classic compatible installers with StuffIt InstallerMaker.

Prices start at $250

To learn more, visit: www.stuffit.com/installermaker/

StuffIt InstallerMaker™

The complete installation solution.™

Aladdin Systems

www.stuffit.com
(831) 761-6200

© 2002 Aladdin Systems, Inc. StuffIt, StuffIt InstallerMaker, and StuffIt Engine SDK are trademarks of Aladdin Systems, Inc. The Aladdin logo is a registered trademark. All other products are trademarks or registered trademarks of their respective holders. All Rights Reserved.

Apache web server, which runs our PHP code, which connects to our MySQL installation.)

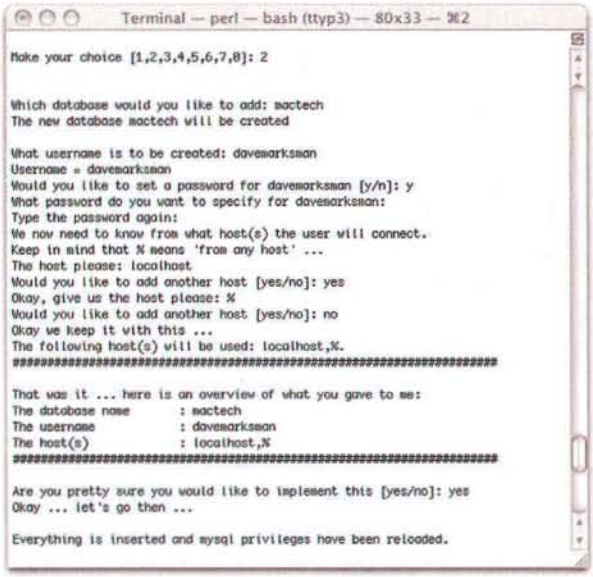


Figure 5: *Adding a new MySQL database and user.*

By default, the user that is created will have very slim access to the specified database (they'll only be able to read data with SELECT, not modify or delete). Since our next article will talk about inserting, modifying, and deleting, we'll want to next choose option 5, which allows us to give davemarksman heightened privileges (**Figure 6**).

Figure 6: *Choosing the database to give heightened user privileges to.*

Once you've created your database and configured your user permissions, you can exit the program by choosing option 0, which will send you back to the shell prompt. You can confirm your database has been created by running /Library/MySQL/bin/mysqlshow –u root –p, which gives output something like:

```
~ > /Library/MySQL/bin/mysqlshow --user root -p
Enter password:
+-----------+
| Databases |
+-----------+
| mactech   |
| mysql     |
| test      |
+-----------+
```

Homework Malignments

I shall no longer prophesize about what will be in the next article, as I seemingly always overshoot my estimates. Instead, students may contact the teacher at morbus@disobey.com.

When you're in the Terminal, there are two environment variables that help control how much typing you have to do: PATH and MANPATH. The first controls what directories will be looked into when you attempt to run a binary program without a full path (like vi compared to /usr/bin/vi), and the second determines what directories are looked into for manual pages. You can see their current configuration by typing printenv in your Terminal. To make working with the MySQL shell programs easier, you'll want to add /Library/MySQL/bin to your PATH, and /Library/MySQL/man to your MANPATH. In Panther, which uses the bash shell by default, add PATH="$PATH:/Library/MySQL/bin" to your (possibly non-existent) /Users/username/.bash_profile. Then, with an authenticating text editor, add OPTIONAL_MANPATH /Library/MySQL/man and OPTIONAL_MANPATH /man to the /etc/manpath.config file. With that finished, restart your Terminal, and you should be able to type man mysqlshow.

You can also add export FTP_PASSIVE=1 to the .bash_profile (see previous Malignment). This instructs CPAN (specifically, the Net::FTP Perl module) to use passive file transfer mode which, for some, is required when the Panther firewall is enabled. Thanks to the macosx@perl.org mailing list for that tidbit.

By David J Harr, Long Beach, CA

OpenGL: An API for Interactive 3D Graphics

3D for fun, profit, and world domination

Introduction

This is the first in an open-ended series of articles intended to explain the basics of 3D interactive programming on the Macintosh using OpenGL. This article will serve as an introduction to OpenGL, give the history of its development, discuss the details of its architecture, and talk about Apple's implementation. Finally, we will close with a program showing some of the capabilities of OpenGL, without going too deeply into the details of the code; it is primarily designed to whet your appetite for what is to come. Future articles in the series will discuss basic topics in 3D programming, including how to construct and display objects, positioning the camera, and texture mapping, just to name a few. After covering the basics, we will go on to more advanced topics such as bump-mapping, shadow volumes, and particle systems. We will begin each article with a short discussion of the mathematics of the topic, then give a basic implementation in OpenGL. If space permits, we may also show some variations on the theme. Finally, we will close with some ideas for advanced experimentation. Hopefully, through these articles, the readers can get a taste of what 3D programming is like, and hone their skills in OpenGL.

Why OpenGL?

Have you ever dreamed of writing a 3D action game? Do you crave the thrill of writing your own flight simulator? Do you live and breathe vectors, matrix math and trigonometry? Then you need to be able to program your computer to display 3D graphics. It was not too many years ago that anyone wanting to do any 3D on a Macintosh or PC was pretty much forced to write all their own routines for doing the graphics. Recently, however, there has been an explosion of interest in the field of graphics, and 3D graphics in particular. In response to this interest, hardware manufacturers such as ATI and nVidia have produced ever more powerful graphics accelerators, usually containing extensive support for hardware acceleration of common 3D graphics operations. Initially, in order to take advantage of the features of these boards, a program had to be specially modified to support each flavor of accelerator. Finally, the people programming 3D applications rebelled, and hardware independent interfaces for the most common graphics boards began appearing.

Microsoft created DirectX, which provided a system level interface for doing graphics operations, and takes advantage of whatever hardware acceleration is available. The problem with DirectX is that it is only available on machines that are running some version of Microsoft Windows. This was not terribly useful for anyone programming on a Macintosh.

Fortunately, there exists an alternative to DirectX that is almost as well supported under Windows as DirectX, and also runs under Linux, most commercial versions of Unix, and Mac OS 9 and X. This is OpenGL. OpenGL is used by many commercial games, including the upcoming *Doom III*, all versions of *Quake*, *Half-Life*, *SpecOps*, and others. It is also used in many high-end rendering and modeling packages such as *Maya*, *3DS Max*, and *Lightwave 3D*. In other words, it is an industrial strength, cross-platform interface to allow programmers to write interactive 3D applications.

History

In 1992, Silicon Graphics (SGI), a maker of high-end graphical workstations, proposed to create an open interface to graphics hardware for use by application programmers. Their proposed Open Graphics Library, or OpenGL, was based on their proprietary IRIS GL library that was used to program their graphics workstations. Drawing on their extensive experience with IRIS GL and graphics programming in general, SGI made significant changes to the IRIS GL to make it more appropriate for use as a general-purpose graphics interface, and presented it as the OpenGL specification, version 1.0. In order to ensure that OpenGL remained an open standard, SGI surrendered control of the specification to the OpenGL Architecture Review Board, a group made up of members from many of the leading

David Harr (and the voices in his head) has been programming Macs and things not Macs more years than he cares to count. As punishment for his many sins, he has been forced to program 3D graphics over and over until he got it mostly right. Currently, he is taking a sabbatical from programming professionally and is enjoying the quaint customs of the natives of Academia. He can be reached at djharr@ucla.edu.

graphics vendors. The ARB approves OpenGL features and extensions, and determines how various implementations need to conform to the published standard. The current version of the OpenGL standard is version 1.4.

OpenGL came to be widely adopted; it became the standard 3D API under X Windows and many versions of Unix, and most video card manufacturers have drivers for Microsoft Windows that support both DirectX and OpenGL. Around the time of the initial release of Mac OS X, Apple decided that they too would support OpenGL, and announced that it was the system interface for 3D rendering on the Macintosh. With the release of Jaguar (Mac OS X version 10.2), Apple extensively revamped its implementation of OpenGL, using it as the basis for many of the graphics capabilities for the Core Graphics, bringing sophisticated graphical operations such as transparency, compositing, rotation, and scaling to the standard imaging model, greatly increasing the range of graphical capabilities of even the simplest applications.

Architecture

OpenGL is an API that is designed to provide device independent access to a common core of 3D graphics capabilities, while still allowing for hardware acceleration where the hardware supports it. In addition, there are extensions allowed to the core capabilities that can allow the programmer access to hardware specific features. OpenGL is a real-time graphics API; it is designed for interactive 3D graphics, not for off-line rendering. Therefore, OpenGL is designed to function as efficiently as possible. To this end, OpenGL functions at a lower rather than a higher level to allow for efficient implementations. Objects are defined as points, lines, and polygons, rather than as cubes, spheres, or other higher level objects. Lights are specified per vertex, rather than using a sophisticated mathematical illumination model such as ray tracing or radiosity.

OpenGL has two main parts. The GL library contains the core routines for interfacing with graphics hardware and performing basic graphical operations. GL contains several hundred commands: commands for the specification of two and three-dimensional objects, as well as commands that control how these objects are rendered into the *frame buffer*, or *video memory*. These commands are implemented as function calls to GL. A program using GL will typically open a window referencing the frame buffer where the objects are to be drawn, then make some calls to allocate and initialize a GL drawing environment or *context*. Once the context is prepared, the program begins making function calls to GL in order to issue commands. There are commands to draw objects made up of geometric primitives such as points, lines, and triangles. Other commands alter the way these objects are drawn, including such things as the color of the objects, whether they are lit or not, the kind of shading they are drawn with, and the way in which the objects are transformed from their own two or three-dimensional space onto the screen. There are also commands to read data directly to and from the frame buffer, for example, copying portions of the frame buffer into *texture memory* (an area of video memory used for storing pixel information to be used in texture-mapping operations) or overwriting the frame buffer with static pixel data. Most of the time, GL operates in *immediate mode*, where issuing a command causes it to be executed. It is also possible to accumulate GL commands into a display list for later execution, and send them down to the GL library all at once. This is referred to as *batch mode* or *retained mode*.

On top of GL is the GL Utility library, or GLU. It uses GL commands to allow the program to operate at a higher level of abstraction than is supported by GL. GLU provides facilities for performing some useful operations. GL is only capable of drawing convex polygons, or polygons that have no holes or indentations in them. GLU can take a non convex polygon of arbitrary complexity, and reduce it to a series of convex polygons, that can be drawn by GL. This process is known as tessellating a concave polygon. In addition, GLU can return the boundary of such a polygon as a series of line segments. GLU also allows you to specify quadrics, such as spheres, cones, and cylinders. GLU then generates the GL primitives to draw these into a display list, so they can be rendered by the application. Finally, GLU allows curves and surfaces to be represented mathematically. GLU supports several popular mathematical representations, including Bezier curves and surfaces and Nonuniform Rational B-Splines or NURBS. Similarly to quadrics, when GLU encounters curves and surfaces of these types, it converts them into a display list of GL primitives for rendering and display. Thus, using GLU, an application can work with 3D graphics at a much higher level than is possible using just the basic GL interface.

Many versions of OpenGL also include a third component, the GL Utility Toolkit or GLUT. GLUT was originally written for X Windows, and has since been ported to most operating systems that have OpenGL available. GLUT provides a platform independent way of handling window and event management, freeing the programmer to concentrate on the rendering portion of the application, rather than worrying about the mundane business of tracking mouse and keyboard events, and handling window update events. Anyone planning to do a cross-platform OpenGL application should consider using GLUT to simplify interface issues. Even if you are only writing for one platform, using GLUT can considerably simplify your application. I will be using GLUT for several of the applications in this and future articles.

ENGINEERING BUSINESS SOLUTIONS

- SINCE 1989 -

PREMIER REPORT SOLUTIONS FOR MAC OS X

VVI®

www.vvi.com info@vvi.com 888-VVI-PLOT

Copyright 1993-2003 VVimaging, Inc. All Rights Reserved. VVI is a registered trademark of VVimaging, Inc. (VVI). Vvidget and Peer Visual are trademarks of VVimaging, Inc. Mac is a registered trademark of Apple Computer, Inc.

Operation

OpenGL is built on a client-server model. In other words, the OpenGL application (client) sends commands to the OpenGL renderer (server). These commands are then interpreted by the renderer, and the renderer modifies the frame buffer in accordance with the commands. Although all implementations of of OpenGL are guaranteed to support the required features of the language, there is no guarantee of the performance or final rendering results of any given command. Increasingly, computers have dedicated graphics hardware that allow for acceleration of many of the more common graphics operations. A good implementation will take advantage of the hardware, but may have to fall back on software rendering for less common cases, which will have much lower performance than the accelerated operations. In addition, there may be mathematical operations on the data that are capable of being accelerated at the cost of some precision in the calculations. For these and other reasons, the OpenGL standard does not dictate the implementation of operations, rather describing the ideal behavior and specifying the range of deviation allowed by implementations. In those cases where deviation from the ideal occurs, OpenGL specifies the rules the implementation must follow to approximate the ideal behavior. Because the behavior of operations is not exactly set out, two different implementations of OpenGL with identical frame buffer configurations may not produce pixel identical results for identical command inputs.

The results of rendering commands sent down to GL are determined by the settings of the current GL *context*. A GL context encapsulates all the state settings of the current drawing environment. These settings consist of values like the current lighting model, the current background color, the current texture and texture coordinates, and many others. The graphics context is all these state variables taken together. The values of these state variables are set in the current context by issuing commands to the GL libraries through function calls. Rather than having to specify a complete set of states every time a piece of geometry is to be drawn, it is possible to set up multiple contexts and switch between them. For example, let us say that an application has two contexts, one context with the settings so that everything is drawn as a wire frame, and another set up so that everything is drawn texture-mapped. If the application keeps references to both contexts, it will be possible to alternate wire frame drawing with texture-mapped drawing merely by switching contexts between objects. OpenGL commands are always processed in the order they are received, so when drawing two objects, all the drawing operations for the first object are guaranteed to complete before any of the second object's commands are executed. One of the results of this is that any queries of the internal state of the context and the pixel values of the frame buffer are guaranteed to be consistent with all the previously dispatched commands executing before any query or pixel operation returns.

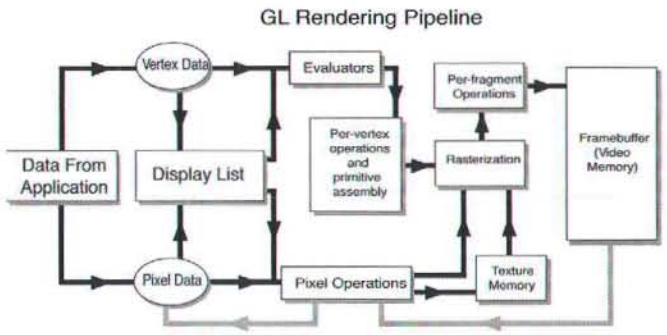


Figure 1. *OpenGL Command Flow Diagram*

An understanding of the GL renderer operation can be helpful in comprehending the results of OpenGL commands. Figure 1 shows the data flow in the GL graphics pipeline. Commands issued by the application enter the pipeline from the left. Looking at figure 1, we can see that a command takes one of two paths, depending whether it deals with vertex or pixel data. Vertex commands follow the upper path. First, they enter the evaluator stage of the pipeline. Here, evaluators provide the means for specifying a polynomial or rational polynomial mapping to produce vertex coordinates, normal coordinates, texture coordinates and colors. These values are then passed along to the later stages of the pipeline as if they had been provided to the pipeline directly by the client. Evaluators are the mechanism that GLU uses to create GL vertex data from Bezier and NURBS surfaces. Vertex primitives (points, line segments, and polygons), are operated on in the per-vertex operation phase. Here, vertices are transformed and lit, and primitives are clipped to the viewing volume. In the rasterization stage, the rasterizer analyzes the vertex data and produces a series of frame buffer addresses, known as fragments. Each fragment is then fed into the last stage of the pipeline, per-fragment operations. This stage performs any final operations on the fragment before it is stored as a pixel in the frame buffer. Among the operations performed are conditional updates to the frame buffer based on the state of the depth buffer, blending the fragment color value with the current pixel value in the frame buffer, subpixel sampling for antialiasing, and various arithmetic and logical operations on the pixel values in the frame buffer.

Commands dealing with pixel data bypass all the geometric operations and are instead processed as pixels directly in the pixel operations stage. Results are then stored as texture memory for later use in the rasterization stage, or are rasterized. In the case of rasterization, the resulting fragments are stored to the frame buffer just as if they were generated and rasterized from geometric commands in the

other command path. Once pixels have been written to the frame buffer, they can be copied back to the pixel operations stage, and then either used as textures or sent back through the pixel pipeline for further processing.

Figure 2. *Screenshot of FirstGL*

Putting It All Together - FirstGL

We have covered the basics of the architecture and operation of OpenGL. "How do you write a program with it?," I can hear you asking. This month's article comes with the source code for a very simple OpenGL program that simply opens a window and draws a cube with six colored faces rotating in space. When the mouse is clicked in the window, the rate of rotation of the cube changes. A click in the left side of the window will change it in one direction and a click in the right side of the window will change it in the other. A command-click or right mouse click will switch it from drawing filled to drawing wireframe. We will briefly examine the structure of the program and the sorts of GL commands that it uses to do the drawing. However, an in-depth discussion of all the techniques and a detailed explanation of all the concepts will have to wait for a later day. Everything that this program does will be explained in mind-numbing detail in the coming months.

Listing 1: main() (FirstGL.c)

int main(int argc, const char *argv[])

Sets up the initial GLUT environment and registers the callbacks we will be using in the program. The only callbacks FirstGL uses are the mouse, idle and display callbacks.

```
// forward declarations of the callbacks
void init (void);
void idle (void);
void display (void);
void mouse (void);

// our main function
int main (int argc, const char * argv[])
{
    // initialization for the GLUT library
    glutInit(&argc, (char **)argv);

    // We are using an RGB, double buffered window, with a z-buffer
    glutInitDisplayMode(GLUT_DOUBLE |
                        GLUT_RGB | GLUT_DEPTH);

    // top left corner of the window
    glutInitWindowPosition(WINDOW_X, WINDOW_Y);

    // Make a WINDOW_WIDTH X WINDOW_HEIGHT window
    glutInitWindowSize (WINDOW_WIDTH, WINDOW_HEIGHT);

    //The string specifies the title of the window
    g_window = glutCreateWindow("FirstGL");


    // this is the function where we do our initial OpenGL setup. Note
    // that it is called AFTER the window is created. OpenGL will only
    // function when the OS has already set up the windowing environment.
    init();


    //The functions calls below all install event callbacks into GLUT.
    //They specify the functions that GLUT needs to call whenever events
    // of a certain type happen. We will only use the idle, display, and
    // mouse click functions, although there are also provisions for mouse
    // moved, keyboard, window resizing and other callbacks as well.


    // the idle function updates the rotation of the cube, and forces a redraw
    glutIdleFunc(idle);

    //This function does the heavy lifting for the drawing in the window.
    glutDisplayFunc(display);

    // when the mouse is clicked in the window, the speed of the rotation changes
    // on command/right-clicked it switches to and from wireframe
    glutMouseFunc(mouse);


    //This function will never return. Whenever it encounters an event, it
    // calls the appropriate callback. It will also quit the application.
    glutMainLoop();
    return 0;
}
```

In order to simplify the programming, FirstGL uses the GLUT library for all window and event handling. So, the first thing to do is to look at how FirstGL's program code interacts with GLUT. In the main function, FirstGL makes a series of calls to functions of the form glutXXXXXX. Unsurprisingly, these are GLUT library functions. First, glutInit is called with the command line arguments that were received by main. Then, FirstGL calls glutDisplayMode with the arguments GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH. GLUT_DOUBLE tells GLUT to make the context double-buffered. In other words, FirstGL will be drawing into one buffer, while the other is being displayed. This is a way to reduce flicker while doing animation. GLUT_RGB tells GLUT that to create a window using RGB coloring, instead of indexed colors, so no palettes to worry about. GLUT_DEPTH instructs GLUT to allocate a depth buffer (or z-buffer), which is a method for drawing correctly depth sorted objects.

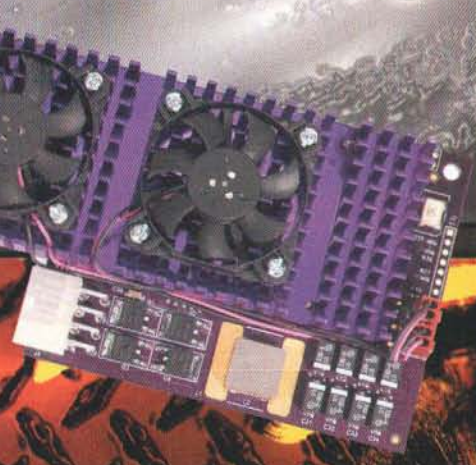
The next three calls to GLUT, glutInitWindowPosition, glutInitWindowSize, and glutCreateWindow are reasonably self-explanatory. The next function main calls is the first of FirstGL's own functions, the init function. This is where FirstGL sets up the

DEV DEPOT®

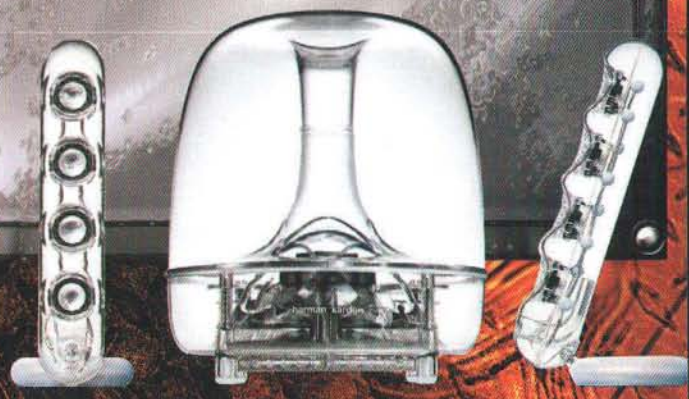
877-DEPOT-NOW

Ready For An Upgrade? DevDepot sells all the Tools and Toys To Max Out Your Mac!

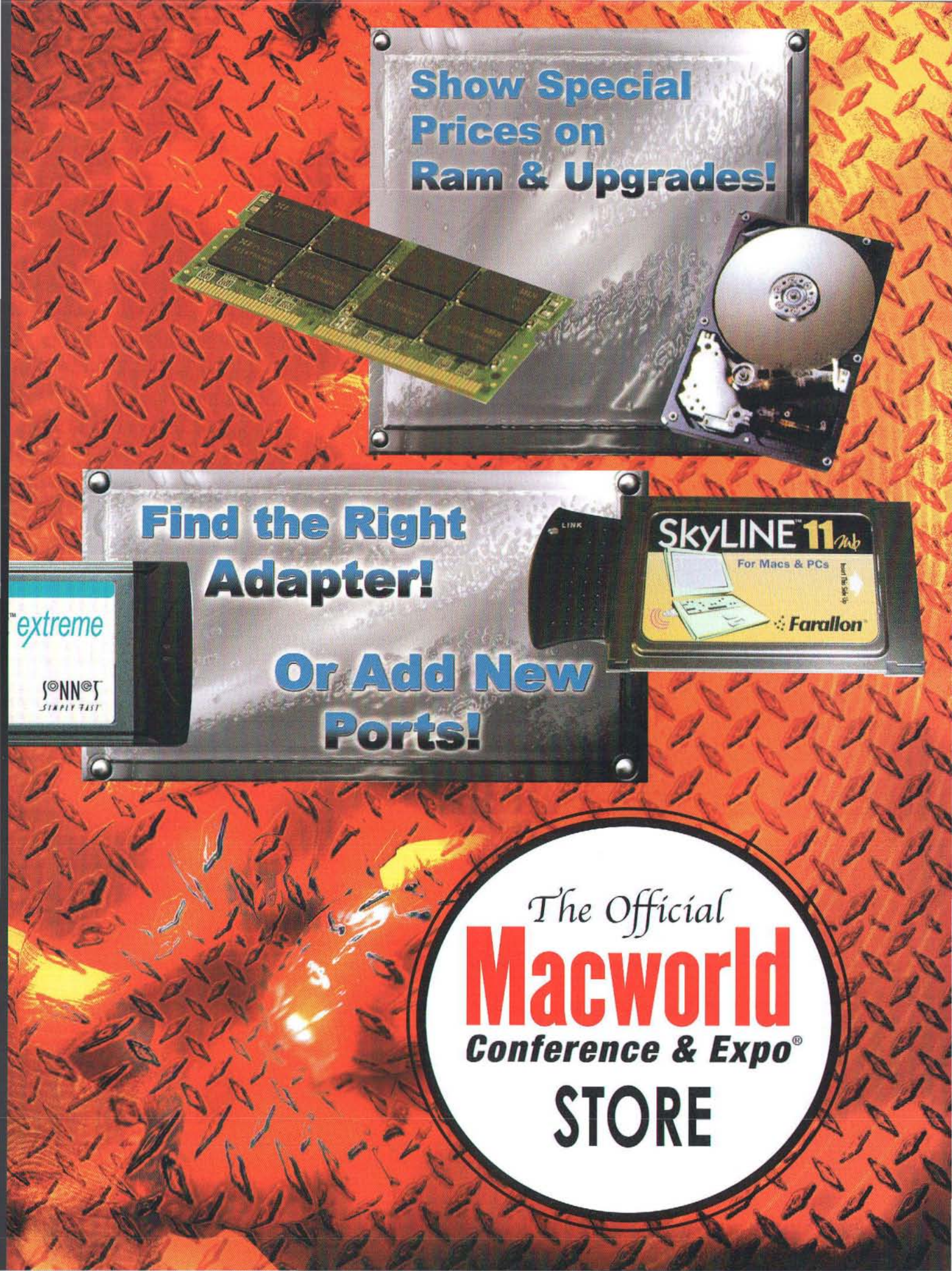
Best Deals on Processor Upgrades



Pump Up the Volume Audio & iPod Upgrades!



www.devdepot.com/bettermac


Show Special Prices on Ram & Upgrades!
Find the Right Adapter!
Or Add New Ports!
extreme
SONNET
SIMPLY FAST
LINK
SkyLINE 11
For Macs & PCs
Farallon
The Official
Macworld
Conference & Expo®
STORE

OpenGL environment. We will examine that code a bit later. The next three calls, to glutIdleFunc, glutDisplayFunc, and glutMouseFunc, register callbacks for idle events, display events, and mouse click events, respectively. Finally, FirstGL calls glutMainLoop, which is GLUT's main event handler. From this point on, the code runs completely inside glutMainLoop, returning only when the application quits.

From within glutMainLoop, GLUT makes calls to the callback functions that have been registered with it. Each time through the main loop, the idle function is called, then GLUT polls an internal event queue. For every type of event, if the application has registered a callback for that event type, GLUT calls the callback routine. FirstGL registers callbacks for mouse-clicks, window updates, and idle events. Other possible events that an application can register callbacks for are mouse-moved, keyboard, window resizing, and window moving, as well as other, less common types of events.

The way that FirstGL works is as follows. In the idle callback, the rotation of the cube is modified by the amount stored in rot_change. Then, the idle callback calls glutPostRedisplay, which forces a redraw of the window. If the user clicks in the window, a click in the right half of the window increases rot_change, and a click in the left half of the window decreases it. When rot_change becomes negative, the direction of rotation reverses. Finally, when the idle function calls glutPostRedisplay, the display function is called, and that is where all the actual drawing commands for the cube are issued.

Listing 2: init() (FirstGL.c)

void init(void)

Sets the OpenGL environment variables to the desired initial state.

```
// this is the initialization function. Here, we set all the
// initial parameters for the OpenGL drawing environment.
void init (void)
{
   // enable depth buffer, so that drawing is depth sorted correctly
   glEnable(GL_DEPTH_TEST);

   // Allow a more realistic shading model
   glShadeModel(GL_SMOOTH);

   // all vertices for faces go in counter-clockwise order
   glFrontFace(GL_CCW);

   // set the background color to white
   glClearColor(1.0, 1.0, 1.0, 1.0);

   // turn on lighting in OpenGL
   glEnable(GL_LIGHTING);

   // turn on light 0 (out of 8)
   glEnable(GL_LIGHT0);
}
```

There are two places where actual OpenGL calls are made. One is in the init function, and the other is in the display function. Let's look at init first. init() is composed entirely of commands that modify the settings of some of OpenGL's state variables. In other words, these calls put the OpenGL context into the state the application wants for the drawing to look the way it expects it to. The first call is to glEnable(GL_DEPTH_TEST). The function call glEnable allows an application to enable and disable a wide variety of features in OpenGL. In this case, the parameter of GL_DEPTH_TEST instructs OpenGL to start using the depth buffer that we told GLUT to allocate in our call to glutDisplayMode back in main. The call to glShadeMode(GL_SMOOTH) informs GL that FirstGL wants all the polygon faces to be realistically shaded. glFrontFace(GL_CCW) tells the renderer to expect all geometry to be constructed such that the front of the polygon is defined by the face formed by following the vertices in counter-clockwise order. glClearColor tells OpenGL to erase the window to white. Finally, init has two more calls to glEnable. In the first one, it activates lighting in the scene. In the second one, it activates one of the eight OpenGL lights. If init didn't activate a light, the cube would be drawn as a black solid, even though lighting was enabled in the scene.

Listing 3: display() (FirstGL.c)

void display(void)

Called whenever there is a redraw event for GLUT. This is where all the actual drawing in the program takes place

```
// this is the callback for window drawing - the meat of the program
// lives here. Everything else is just window dressing, pardon the pun.
void display(void)
{

   <Camera and window setup omitted for clarity>

   // start transforming the model space
   glMatrixMode(GL_MODELVIEW);

   // initialize the matrix stack to the identity matrix
   glLoadIdentity();

   // set the rotation for the cube, values set in the idle function
   glRotatef(rot[0], rot[1], rot[2], rot[3]);

   // cant the cube on its end, 45 degrees around the x-axis
   glRotatef(45.0f, 1.0f, 0.0f, 0.0f);

   // from here, we are specifying the vertices of the faces of the cube.
   // as we stated in the init function, the vertices are in CCW order.
   // the normals are the same as the vertices, since the center of the
   // cube is at the origin.

   // Each face is wrapped in a glBegin/glEnd pair.
   glBegin(GL_QUADS);
   glColor3f(1.0f, 0.0f, 0.0f); // red face
   glVertex3f(1.0f, -1.0f, 1.0f);
   glNormal3f(1.0f, -1.0f, 1.0f);
   glVertex3f(1.0f, -1.0f, -1.0f);
   glNormal3f(1.0f, -1.0f, -1.0f);
   glVertex3f(1.0f, 1.0f, -1.0f);
   glNormal3f(1.0f, 1.0f, -1.0f);
   glVertex3f(1.0f, 1.0f, 1.0f);
   glNormal3f(1.0f, 1.0f, 1.0f);
   glEnd();

   <Repeat five times, one for each face of the cube>

   // start displaying the buffer we have just finished drawing into.
   glutSwapBuffers();
}
```

Now, let us look at the heart of the program, display. At the top of the function is a bit of bookkeeping to get the correct camera position and transformation matrices in place. Again, all these commands are commands that modify the current setting of the GL context. None of them actually cause any drawing to take place. Finally, comes a call to glBegin(GL_QUADS). This is where drawing

starts taking place. Most drawing commands in OpenGL are contained in a glBegin/glEnd block. glBegin(GL_QUADS) informs OpenGL that a series of vertices and possibly other information is coming down, and that these vertices make up a quad, or 4 sided polygon. After the glBegin, display sets the color for the polygon. Then, display starts specifying the coordinates for the points, and the direction of the normals, for color calculations. Since the center of the cube is located at the origin, the direction of each normal is simply the coordinates of the associated vertex. display specifies 6 faces for the cube. OpenGL applies the transformations that were set out at the beginning of display, draws the transformed polygons, colors them, and then swaps the buffer so that the finished drawing is displayed in the window. And that is essentially how a program uses OpenGL. The drawing environment is controlled through commands that set the values of the various state variables of the context. The appropriate transformations are sent down, then geometry is specified, and finally, the finished drawing is shown.

Acknowledgements and Further Reading

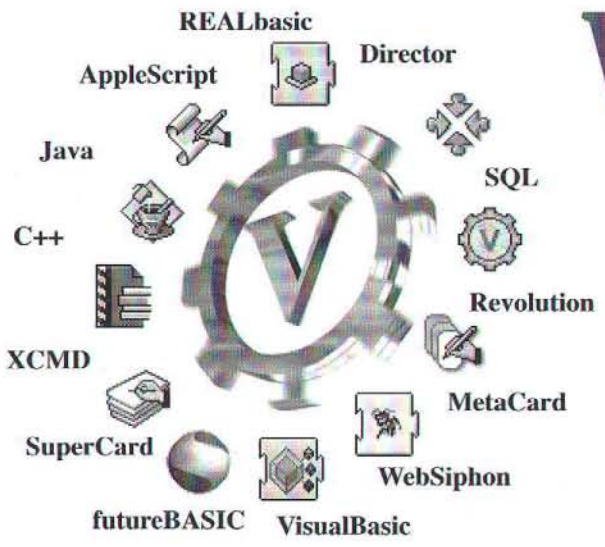
Many thanks to my wife, Carol, who suffered through several drafts of this article, each seemingly more impenetrable than the last, but who finally got me to write it in such a way that she could follow my meandering exposition. Thanks also to Paul Snively, who provided important feedback on the direction I was going with the article, and also proofread it, pointing out several errors. He also did the OCaml version of FirstGL. Also not to be forgotten is Chris Page, who looked over the article, and translated FirstGL to Dylan. Finally, thanks to Marshall Clow, who first gave me the idea of writing these articles, and then kindly led me through the process of getting it in to the folks at MacTech.

You may notice that I mention an OCaml and Dylan version of FirstGL. Undoubtedly, you are wondering what I am talking about. OCaml and Dylan are both programming languages that take a very different approach to the C/C++ family of languages. It is almost certain that all of us would be a lot more productive and write more correct code faster if we were using these languages to develop in. In an attempt to get people to look at some interesting alternative languages, I am going to try to provide versions of all the applications for this series in C/C++, Ocaml and Dylan. To find out more about OCaml, you can go to http://www.ocaml.org. You can learn more than you ever wanted to know about Dylan at http://www.gwydiondylan.org. I encourage you to check both languages out, if for no other reason than to become acquainted with a different programming worldview.

Anyone who is serious about programming OpenGL cannot live without the OpenGL Programming Guide or the OpenGL Reference Manual. My discussion of the architecture and operation of OpenGL was heavily influenced by both of these, and the illustration of the OpenGL command flow diagram in figure 1 was inspired by an illustration in the Programming Guilde. Both of these books have a wealth of information in them regarding just about every facet of OpenGL you could care to name, although it is not always completely obvious where things are at times. There are also numerous web pages and various discussion groups on the web for OpenGL programming. A good place to start would be the official OpenGL web site: http://www.opengl.org.

OpenGL Architecture Review Board, et al. *OpenGL Programming Guide, The Official Guide to Learning OpenGL, Version 1.2.* 3d edition. Addison-Wesley, Reading, Massachusetts, 1999.

OpenGL Architecture Review Board, Dave Shreiner, ed. *OpenGL Reference Manual, The Official Reference Document to OpenGL, Version 1.2.* 3d edition. Addison-Wesley, Reading, Massachusetts, 2000.



Valentina

Object-Relational SQL Database

The fastest database engine for MacOS/Windows

It operates 100's and sometimes a 1000 times faster than other systems

www.paradigmasoft.com

Hosted by macserve.net

Download full featured evaluation version

By John C. Welch

Panther

A look at the latest release of Mac OS X

Welcome

On October 24th, 2003, Apple released the latest version of Mac OS X, aka Panther, (continuing in Apple's feline naming trend), version 10.3. This version boasts not only the changes we've all seen in demos, such as the new Finder, and Expose, but some other, less visible features that are just as important.

Overview

With any major operating system release, there are a lot of changes that the users, developers, and administrators get hit with, and Panther is no exception. While the release may be numbered as a tenth version change, this could easily almost qualify for a full version change. While Apple is always running the hyperbole machine non-stop for any major release of Mac OS X , in this case, it's more justified than usual. There are huge amounts of changes, inside and out, and they make for one heck of an upgrade.

Now, there have been numerous reviews of Panther already, so thanks to them, there are a few things I'm not going to bother with. Expose, the new appearance of the Finder, both have been done to death. We're not ignoring the Finder, but there have been forests published on the new appearance, and from my POV, that's enough. I care less about the view, and more about the features, and those, I will look at. Fast User Switching, FUS, is going to be looked at, but not for the pretty transitions. So, the features that I look at are going to be those of interest to me as a network administrator / IT Geek. While I will be mentioning Mac OS X Server, it's only going to be in relationship to Mac OS X, not as a separate review. I'm not going to really talk about iApps, WebObjects, or the developer tools. Those aren't really a part of the core OS. I'm also not going to spend much time on the BSD improvements, as those are better handled by a separate article, since the BSD layer is its own world almost.

Installation

As with any OS, the first exposure to the product is the installation, and this is the area that Panther has had the most problems. The upgrade installs have been a disaster in most cases. There have been consistent problems with permissions being set wrong, needed system users not being created, etc. At this point, barring a new CD release with 10.3.1 or later on it, I would avoid doing an upgrade install from Jaguar, or any earlier version of Mac OS X. However, that's not to say you have to reformat your system to get a good install of Panther. When you do the install, change your install type from Upgrade to Archive and Install, but pick the option that says it will preserve user and network settings. That way, you don't have to reset all your home directory settings, network settings, etc. You'll end up with a directory called "Previous Systems" on your Mac, and it will have all the non-user stuff in it. Now, don't just delete this. You're going to want to hang onto it for a few days until you're happy that everything is working correctly. I find that if I go into the Library, (not System/Library) directory in Previous Systems, and manually move over the stuff that isn't in my new /Library, I avoid problems. I don't just copy over all the folders in the old /Library, but rather the stuff inside them. This way, any serial numbers, etc. that are in /Library are there in Panther too. (Adobe likes to put serial numbers in /Library/Application Support/, for example.)

Of course, you CAN do a format install, but I've not found it necessary. As you may have heard, you can only install Panther on Macs that shipped with built-in USB. So Beige G3's are out, as are laptops prior to the 1999 PowerBook G3 models. Note, that's not to say it's *impossible* to install on these models, just that Apple's not supporting it. Regardless of model, a poor or inconsistent installation experience is never good, and hopefully, Apple will get new CDs out soon.

Regardless of your installation method, there are some obvious changes. For one, there are now three CDs for Panther alone, not counting the Xcode Development tools CD. (In an interesting difference between the two, Mac OS X Server 10.3 only has two CDs). This is due to overall increased size of the OS, and new items, such as X11. X11, the, well, X11 environment for Mac OS X from Apple is an optional install in Panther. To use it, you

John Welch <jwelch@provar.com> is an IT Staff Member for Kansas City Life Insurance, a Technical Strategist for Provar, (http://www.provar.com/) and the Chief Know-It-All for TackyShirt, (http://www.tackyshirt.com/). He has over fifteen years of experience at making Macs work with other computer systems. John specializes in figuring out ways in which to make the Mac do what nobody thinks it can, showing that the Mac is a superior administrative platform, and teaching others how to use it in interesting, if sometimes frightening ways. He also does things that don't involve computertry on occasion, or at least that's the rumor.

16 ~~15~~ years and running.

Ever since we built our first Ethernet adapter in 1987, we have been devoted to making networking faster, easier, and safer.

Now, our comprehensive line of Gigabit Ethernet solutions makes high-speed networking simpler than ever – at prices below $20 per port.

Mac OS. Windows. Linux. The choice is clear.

Asanté Technologies.

FriendlyNET® GX5-W 10/100/1000 Gigabit Ethernet Smart Switches

NEW!

Deliver Gigabit Ethernet to the desktop with comprehensive management via any web browser!

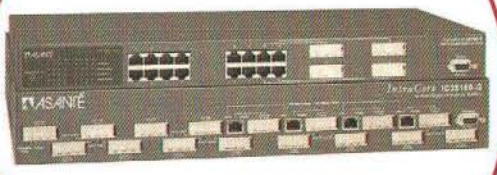
"FriendlyNET GX5-424W ... its price tag is nice and short"
Processor.com

GigaNIX™ – the first Gigabit Ethernet adapter compatible with Mac OS, Windows, and Linux

Gigabit speeds over existing copper wiring!

"Asanté GigaNIX ... an impressive set of features at a great price."
Networks Today

IntraCore® Multi-service Gigabit Ethernet backbone switches



Uncompromised flexibility in any environment!

" Asanté IntraCore ... well suited for handling multi-media traffic including VoIP and video, even in a mixed computing environment."
PC Magazine

Have an older Mac? See asantestore.com for all of your networking needs.

ASANTÉ

1-800-662-9686
www.asante.com

Built for Mac OS X

Microsoft® Windows® compatible

© 2004 Asanté Technologies, Inc. Asanté, FriendlyNET, and IntraCore are registered trademarks and GigaNIX is a trademark of Asanté Technologies. All other trademarks are property of their respective owners. All rights reserved.

have to choose to do a custom install, once you have chosen your installation type. There are other options here besides X11. You can install more printer drivers, more languages, more fonts, etc. For my part, I always install all the printer drivers on a laptop, (you never know where you'll be printing, or to what), however, on a desktop, I don't bother, unless I know I'll need those specific drivers. If you install X11, you'll need to have the third CD at hand. (Of course I install it, I'm a geek).

The install can take quite a while, depending on your circumstances. There's a reboot after the first CD, and the rest of the install runs booted from your hard drive. For Mac OS X Server - based networks, the remote install options with Panther and Panther – Server are much improved over Jaguar. Creating the install images is straightforward, as is setting up to install via NetBoot. Once that's done, just NetBoot the clients from the appropriate image, and go. On a slow network, with an old B&W G3 as a server and a grape iMac, it took about 45 minutes. On a faster network, it took about 20 minutes.

Finally, another neat trick is exposed in Disk Utility. It seems that having Mike Bombich working for Apple has paid off a bit, as it looks like a lot of the features of Carbon Copy Cloner are now a part of Disk Utility, and you can restore a drive from a networked image via WebDAV. Not something I've had time to test, but a pretty cool possibility nonetheless.

Configuration

Machine configuration and System Preferences in Panther are pretty close to what they were in Jaguar, but there are some new features, that were needed, and some reorganizing that makes it much faster to see what interfaces are working on your system. The network preferences got a lot of redesign in particular, and you benefit from that work as soon as you open them.

Figure 1. *Network Status screen*

As you can see, the interfaces you have enabled, and the status of those are immediately shown to you, so if you are having problems, you have an immediate starting point. This is something that all users, but mobile ones in particular have needed for a while. The TCP/IP settings have also gotten some changes, most noticeably the inclusion of a machine's IPv6 address, and the ability to configure IPv6 settings in the GUI.

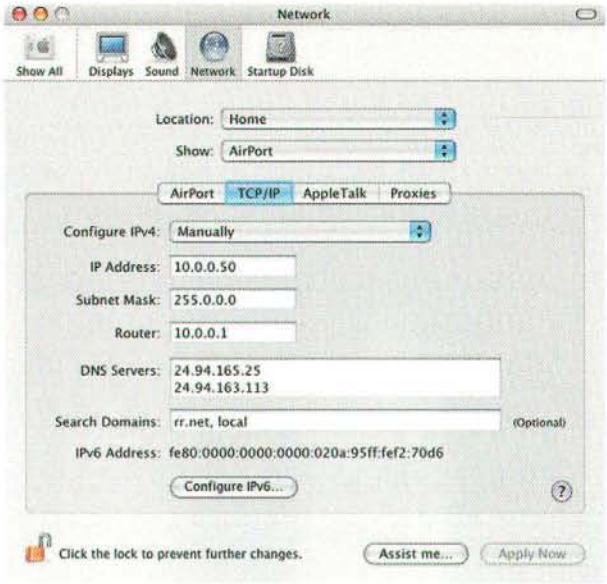


Figure 2. *TCP/IP setup*

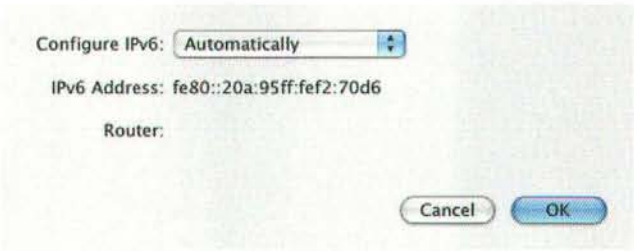


Figure 3. *TCP/IPv6 automatic setup*

As you can see, the default is to automatically set up IPv6 settings. There's a reason for this. Here's what the manual version looks like:

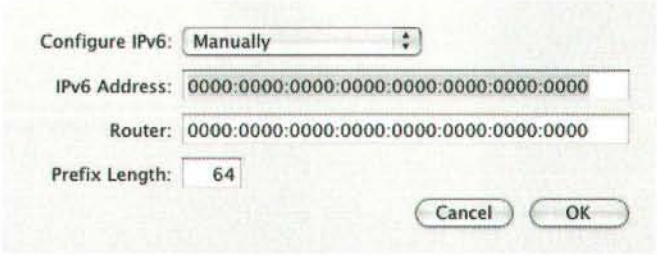


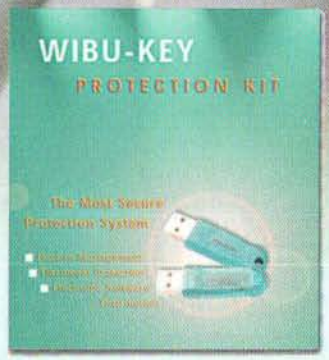
Figure 4. *TCP/IPv6 manual setup*

You thought IPv4 addresses were bad. But the support is not just the base networking stack. Apple has been gradually

WIBU-KEY Software Protection

Be sure to order a **WIBU-KEY Protection Kit** and find out why small software companies to Fortune 500 enterprises are switching to the WIBU-KEY **Security, not Obscurity** model for all of their software licensing needs, including:

- **Common API across all platforms and hardware**
- **Hardware-based code and data encryption**
- **Field-upgradable and reusable hardware**
- **The most cost-effective network licensing solution available**
- **The only system to employ RID/RED and AXAN security**
- **Engineered and manufactured to exacting ISO9001 standards**



Test the
WIBU-KEY
Protection Kit
sales@griftech.com

Call 800-986-6578

The Key is in Your Hands!

WIBU SYSTEMS

WIBU-SYSTEMS USA, Inc.
Seattle, WA 98101
Email: info@wibu.com

www.griftech.com
www.wibu.com

Protection Kits also available at:
Belgium wibu@impakt.be, **Denmark** lean@danbit.dk, **Finland** finebyte@finebyte.com, **France** info@neol.fr, **Hungary** info@mrsoft.hu, **Japan** info@suncarla.co.jp, **Jordan, Lebanon** starsoft@cyberia.net.lb, **Korea** dhkimm@wibu.co.kr, **Luxembourg** wibu@impakt.be, **Netherlands** wibu@impakt.be, **Portugal** dubit@dubit.pt, **Thailand** preecha@dptf.co.th, **United Kingdom** info@codework.com, **USA** sales@griftech.com

implementing IPv6 capabilities in their applications as well, although most are in a transitional state, so will look for IPv4 connections, and tend to prefer them. But for those of you working in pure IPv6 environments, Panther is a much nicer experience than Jaguar was.

Another problem in Jaguar was setting up custom Ethernet settings. Although you could change Ethernet settings via ifconfig, getting those changes to stick was often much harder than it needed to be. Panther allows you to change Ethernet settings in the GUI.

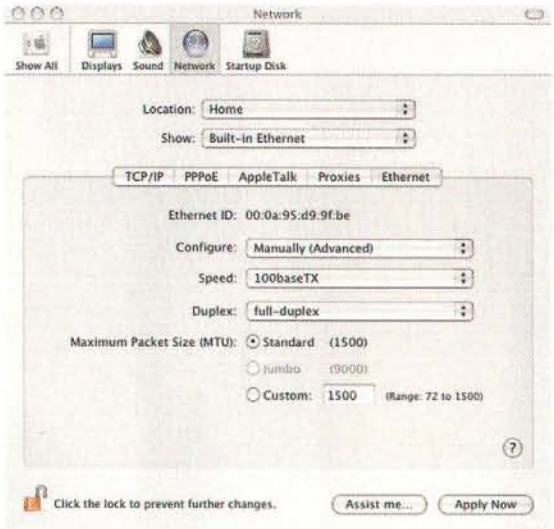


Figure 5. *Manual Ethernet setup*

While you can't set jumbo frames in the UI, (that's limited to Mac OS X Server), for those of you in situations where autoconfig didn't work so well, you can now customize your Ethernet settings far easier than you could in Jaguar.

Printing has made it into System Preferences, and you can now set up a few basic things directly in System Preferences, such as the printer that comes up in the Print Dialog, and the default paper size. (One would think this qualifies as the Default Printer, but evidently Apple disagrees with me there.) You can enable printer sharing, and fire up the Printer Setup Utility, the Application Formerly Known As Print Center, so that you can set up specific printers. (More on that later.) Panther now has built-in faxing, and while it's not a replacement for a fax server, the ability to have your incoming faxes sent to an email address is a nice touch. Support for internet faxing would be a nicer touch, but for a first effort, it's not bad. Unfortunately, you can't easily share the faxing as you can the printing, which is a bit of a letdown.

One not so nice change is the removal of functionality from what was the Internet preference pane, (now the .Mac pane). The .Mac pane now only lets you set your .Mac account information, and iDisk settings. You have to go to Mail, Safari, etc. to set the default web browser, home page, email application, etc. While simplicity is nice, this is more of a crippling effect than a simplifying one, and has a net effect of making it harder to set often – used preferences. Since the default email or web application is a system, or at least a user – wide setting, it makes no real sense to make you go find an email application to set this in. If you can set other user prefs in System Preferences, then removing these just looks like a silly attempt to push people into using Safari and Mail.

Security has gotten new attention from Apple in Panther. It now has its own preference pane, and some new tricks. The big one is FileVault, which turns your home directory into a big encrypted disk image. Problems with FileVault aside, there are some potential issues with it that would make one not want to just enable it by default. If you use AppleScript a lot, FileVault changes the path to your home directory. Because FileVault makes your home directory a single file, and an encrypted one at that, if that file gets corrupted, recovery of data will be almost impossible. Any change to even a small text file can force the backing up of several Gigabytes of unchanged data. FileVault is a great idea if you need it, but understand that there are real issues and problems that you will run into because of it. However, aside from FileVault, there are other, less worrisome security features in Panther. You can finally require a password to wake your computer from Sleep, a feature much in demand by laptop users. Unfortunately, this is tied into the screen saver, so you either password enable both, or neither. Some granularity here would be nice. Both are tied into the new Kerberos security improvements in Panther, so if you are on a Kerberos network, unlocking the screen saver or waking from sleep can be tied into your network authentication system. Another new feature is the idle-time logout, a welcome feature for anyone running a Mac in a lab, or other public use situation.

The user account settings get yet another makeover in Panther. Login items no longer get their own preference pane, they're back to being part of your account settings, which makes sense. The FileVault settings from the "Security" pane are replicated in the Security tab. If you are clicking on someone else's account, and you're an administrator, the Login Items tab changes to a Limitations tab. For people not using Mac OS X Server and Workgroup Manager, this is the place to lock down parts of the OS for standard users, or limit them to the Simple Finder. Clicking on Login Options lets you choose how the login window looks, enable or disable autologin, hide the Sleep, Restart, and Shutdown buttons in the UI, and enable Fast User Switching.

Fast User Switching is simply the ability to switch between users on a machine without logging the current user out. Windows XP does this. You've always been able to do this in the UI from the command line, via the su command. Fast User Switching simply enables this at the GUI level. Now, if the account you are switching to has a password, you'll need to enter that. Certain applications, such as iChat and iTunes don't play nice with Fast User Switching. iTunes simply ignores it, and only runs in one user environment at a time. iChat will switch, but it will log out the switched – out user(s). Some utilities don't work well, or get confused by Fast User Switching, such as TypeIt4Me. I've also found that if you are using the Active Directory plugin, using Fast User Switching to switch to an Active Directory account that is not already logged in will send the GUI south until you reboot, or log in remotely and kill the

loginwindow process for the AD user. Also, the more switched out users you have, the more that Mac's resources will get used. But all in all, it's a good implementation of a requested feature.

The Energy saver finally gets the scheduled shutdown/startup features, missing in Mac OS X prior to Panther. The Keyboard and Mouse preferences have been combined, and a new trick, one that lets you define custom keyboard shortcuts is included. You can also use this to change application shortcuts. For example, the login/logout of AIM shortcut in iChat is Cmd-L. Unfortunately, this is also the "enter URL" shortcut in every web browser on the Mac, including Safari. Since the brushed metal interface makes it hard to see at a glance if a window is active or not, I'm always logging out of iChat when I thought I was entering a new URL. However, a new keyboard shortcut, and BAM, that problem is gone.

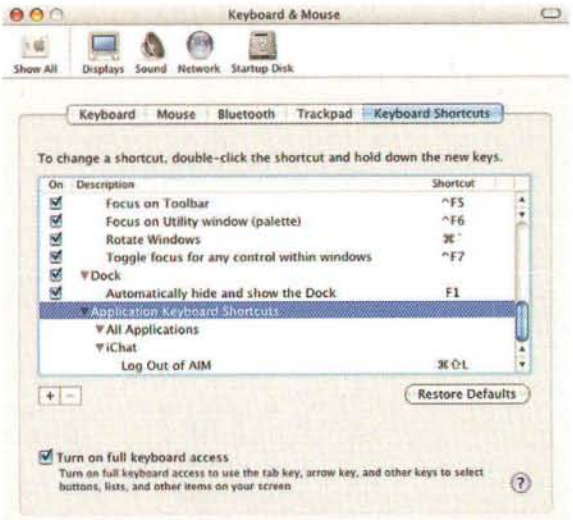


Figure 6. *Keyboard Shortcuts*

There are some limits, for example, I can't override the autocomplete in Script Editor in Panther to be Tab instead of F5 or Option-Esc, because Tab can't be used with this feature, nor can Enter, Cmd-Tab, Cmd-Enter, etc. Still, it's a nice feature, and my one use of it has saved me a lot of frustration. My final favorite new feature in Panther's System Preferences is one that I wouldn't have expected, and that is Classic. Classic now has a menubar widget, and if you have a Classic System folder, this widget also lets you access the Classic Apple Menu in all it's customizable glory. So, in the fourth major version of Mac OS X, Apple has finally given us back the Apple Menu.

There are quite a few configuration changes all throughout the OS. The Apple Menu sports a new "Software Update" item, which is a welcome change. Even better though are the improvements to the command – line softwareupdate utility. With Jaguar, you ran it once with no options to get a list of updates, and then you had to run the command once for each update, which was tedious. Panther improves this greatly.

SOFTWARE LOCALIZATION MADE

easy

POWERGLOT FEATURE HIGHLIGHTS

- LOCALIZE CLASSIC, CARBON™, COCOA® AND PALM OS® APPS
- LEVERAGE EXISTING TRANSLATIONS
- AUTOMATE WITH APPLESCRIPT®
- IMPORT /EXPORT TRANSLATION MEMORIES

www.powerglot.com

browse, translate, click, done.

PowerGlot Software - Localization tools for Mac® OS

www.powerglot.com

info@powerglot.com

Mac OS, Carbon, Cocoa and AppleScript are trademarks of Apple Computer, Inc.
Palm OS is a trademark of Palm, Inc.

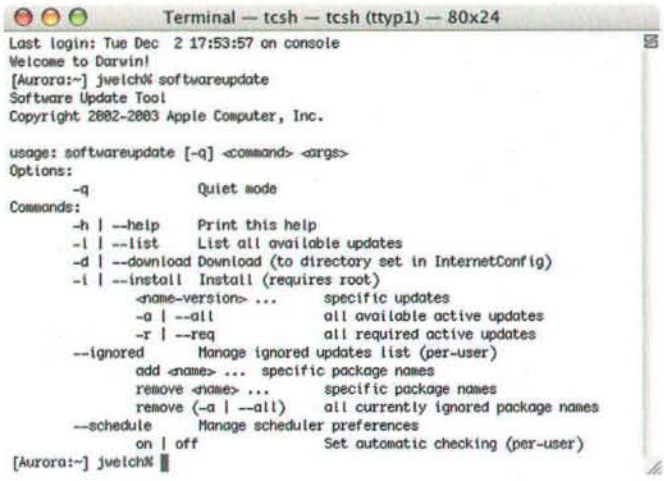


Figure 7. *Command – Line softwareupdate in Panther*

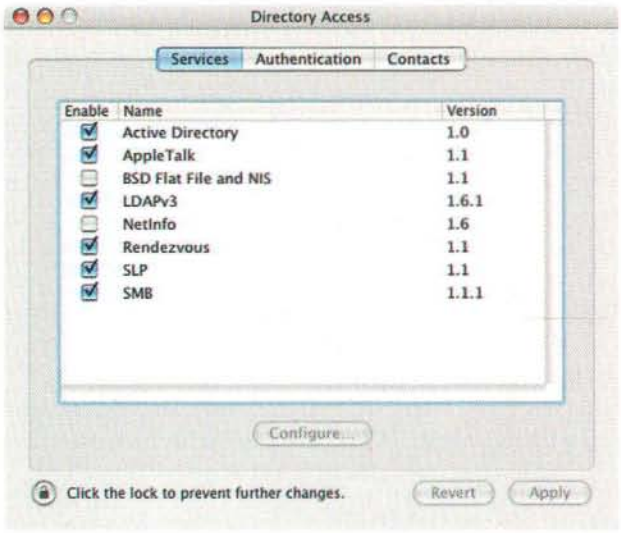


Figure 8. *Directory Access Main Screen*

As you can see, not only can you manage all the different settings of softwareupdate, like schedules, ignored updates, etc. , but you can also elect to install all available updates, all required updates, and, very important for system administrators, you can also download the update packages, a major convenience for those who prefer to roll out updates on their schedule, not Apple's.

One preference pane that is missing is the ColorSync pane. That functionality is now a part of the Displays preferences and the ColorSync Utility. I'm not going to comment on some of the nitty – gritty operations of ColorSync in Panther, I don't know enough about it to do so. But I will say that it is a much bigger part of Panther than it was a part of Jaguar. There are new ColorSync "filters" that allow you to apply ColorSync settings to print files. But it's not just things like embedding profiles. You can also do things like compress images, (a welcome feature for Quartz – created PDFs), sharpen or blur images, create PDF /X – 3 PDFs, etc. ColorSync has gotten a lot of attention in Panther, and for those of you that live and die with it, I highly recommend a bit of research before upgrading, both to prevent problems, and see what the new features can do for you.

Directory Services

The second version of Open Directory has gotten a lot of play from Apple, although for the most part, without Mac OS X Server, most of the changes are lost on the average user. But there are a few changes that you don't need Mac OS X Server to take advantage of. The biggest one is the new Active Directory plugin. This, like all other Directory Services settings, is found in the Directory Access application in /Applications/Utilities.

Now, in addition to Active Directory, there are also plugins for AppleTalk, LDAPv3, NIS and others. LDAPv2 is gone, and NetInfo is no longer enabled by default. Neither is AppleTalk, which means that you are going to have a rough time browsing AppleTalk networks until you enable this, as I have on my machine. This is not a very obvious place to look for this, and the default causes a lot of problems for people, so while I understand that Apple wants to get rid of things like NBP and other non-AFP parts of AppleTalk, they really need to be clearer about showing people how to find this without a trip to Apple's support site.

The BSD Flat File and NIS plugin finally makes plugging into those networks FAR easier than it was, which has been a long time in coming. (I still remember a WWDC Networking Feedback Forum where I asked Apple to either fix NIS, or break it completely, because the 'half-ness' of it was killing me.) It's not going to give you NIS+ connectivity ala Sun boxes, but is you need NIS, it's better now than it used to be. The LDAPv3 Plugin is fairly unchanged from the client point of view, which make sense, as most of the changes with LDAP and Open Directory have more to do with Mac OS X Server than with the clients.

One thing that should be touched on is the recent noise over the 'major' security hole in Directory Services. By default, the LDAP plugin is set to get Directory Information from a DHCP – Assigned LDAP server. This is a part of the DHCP spec, so it's not an Apple-invented trick. The problem is, if you have a rouge DHCP / LDAP server that is set up correctly, it could allow a cracker to take over your system, and that of any other Mac booting on a subnet visible to the rogue server. The problem isn't one that's easily fixable. First, most DHCP security is targeted towards limiting client access to the server, not authenticating the server to the client. That's how DHCP is supposed to work. You find the first DHCP server available, and configure from that. There's no security in this process of any real value. If you grab the wrong DHCP server, you're effectively denied service, or correct service. This is why rogue DHCP servers are "A Bad Thing" anyway. The current DHCP

standard has no security. In fact, the following is the entire security section of the current DHCP RFC (2131):

> 7. Security Considerations
>
> DHCP is built directly on UDP and IP which are as yet inherently insecure. Furthermore, DHCP is generally intended to make maintenance of remote and/or diskless hosts easier. While perhaps not impossible, configuring such hosts with passwords or keys may be difficult and inconvenient. Therefore, DHCP in its current form is quite insecure.
>
> Unauthorized DHCP servers may be easily set up. Such servers can then send false and potentially disruptive information to clients such as incorrect or duplicate IP addresses, incorrect routing information (including spoof routers, etc.), incorrect domain nameserver addresses (such as spoof nameservers), and so on. Clearly, once this seed information is in place, an attacker can further compromise affected systems.
>
> Malicious DHCP clients could masquerade as legitimate clients and retrieve information intended for those legitimate clients. Where dynamic allocation of resources is used, a malicious client could claim all resources for itself, thereby denying resources to legitimate clients.

There is a secure DHCP RFC, 3118, but it's still a proposed standard, not a final one, and has been in the works since 2001. Even after it becomes a standard, creating a secure DHCP infrastructure would require more than a little work at all levels of any network. The fact is, if you use DHCP at this time, you are accepting a certain amount of risk. Apple using a standard in a way that is not against the standard does not suddenly "create" a security hole. As well, if you have to set up a couple hundred machines at once, this out of the box autoconfig ability for LDAP directories that are advertised via DHCP is more than a little handy. Being able to have all your user, home directory, and other LDAP-Derived settings available on a client machine as soon as you power it on is not a minor convenience. So, you have to either give up autoconfig, or keep a closer eye on your network. This is also not a crack that is terribly easy to set up or implement. Rogue DHCP servers get discovered fast, they cause a LOT of problems. In other words, it's something to keep an eye on, but not panic over. Simply disabling the ability to find LDAP servers via DHCP on your client Macs takes care of this hole anyway.

The big new change however is the Active Directory plugin. Now, you could hook Jaguar into Active Directory via LDAP, but this was not a very simple process, and did require some Active Directory schema modifications, which most Active Directory administrators were loathe to make, because it's almost impossible to undo Active Directory schema mods, and because in most large Active Directory setups, the Macs are not

Fetch

Play Fetch with a panther.

X

Fetchsoftworks.com

(Running dog included.)

numerous enough to make that kind of thing worthwhile. So the plugin alleviates much of the pain associated with hooking Macs into Active Directory networks, which is "A Good Thing" for Mac users on Windows networks.

Figure 9. *Active Directory Plugin Setup*

The setup here is straightforward. By default you only have to enter the forest name, the domain name, and the computer ID. (Forest and Domain can be identical). You'll need to know the location within Active Directory the computer is going to be stored. By default, the plugin assumes it's in the "Computers" container, but you can assign it to a different container, or an OU, depending on your needs and your Active Directory setup. You'll need to not only to be able to authenticate as an administrator on the Mac OS X machine, but you'll also need to authenticate with adding machine privileges to the Active Directory domain, or have someone nearby who can. If you are trying to bind remote machines, you can have Directory Access connect to a remote machine, or you can use SSH and the dsconfigad command. "man dsconfigad" for details. The options are pretty straightforward. If you have a laptop, you can create a mobile account with a local home directory, so that you can log in with an Active Directory account offline. You can set it to authenticate in multiple domains in the same forest, a needed feature if you work in a large Active Directory network. You can specify the preferred domain server, and set Active Directory groups that can act as local administrators on a given Mac. Finally, you can map the User ID, UID to a specific attribute.

What does all this mean? Well, you can log in with a valid AD user ID, and you don't have to create it ahead of time. If you make it a mobile account, you get a home directory on the system. Once an Active Directory account has been created on a Mac, it can be set up to be a local administrator for that machine.

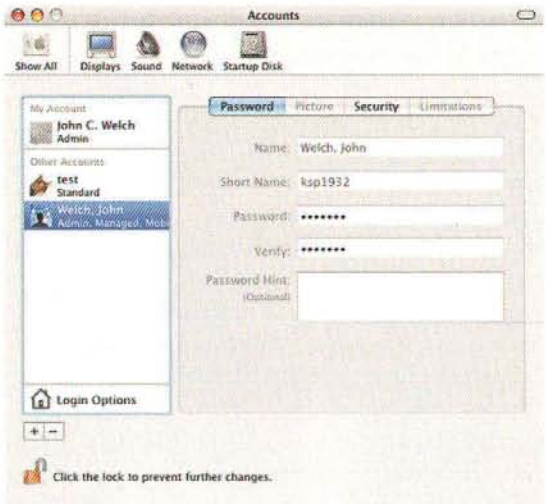


Figure 10. *Active Directory User in System Preferences*

Logging into Active Directory with the plugin gets you Kerberos tickets for Active Directory resources, so you get some single signon benefits. If you want, you can also have Active Directory set up your network home directory so you can use it with Mac OS X. Now, this does not mean that your Active Directory home directory is your Mac OS X home directory. It actually mounts as a separate network drive. But you have access to it. If you set up a custom Contacts search tree in Directory Access, and put the Active Directory node in it, then you get a nice side benefit from Address Book: It can search the Global Address List, or GAL for email and other information. Very nice for Address Book users. If you want better AD integration, such as real SMB home directory support, DFS integration , or you still need to authenticate against NT 4 domains, then your best bet, (and only bet in the case of the NT 4 domains) is Admit Mac, from Thursby Systems. It's more expensive than the Panther plugin, but you get more features too, so its value is determined by what you need.

Other new configuration tricks

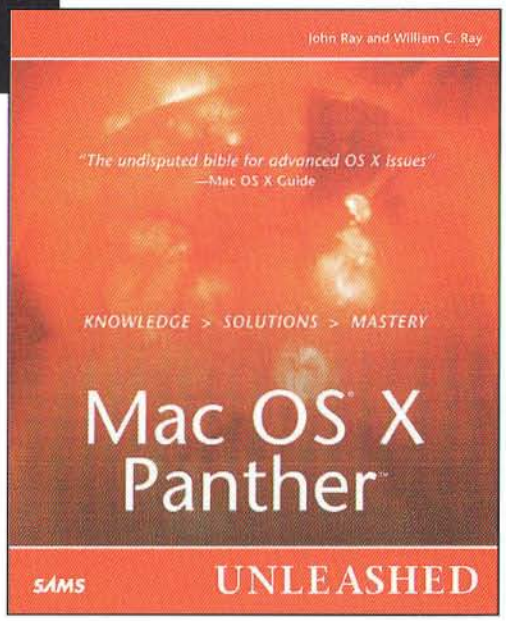
As we've already seen, the command line has been beefed up quite a bit in Panther. Mac OS X Server 10.3 takes this even further, by giving you command – line equivalents for almost every GUI tool. Panther also comes with a copy of the client for Apple Remote Desktop, so if you use that product, you can plug your Panther systems into your workflow right out of the box.

The Finder has some new tricks as well. The obvious one is the return of Labels, which is either a very good thing, if you used them a lot, or a minor thing if you didn't. If you are new to AppleScript, the Finder is now somewhat recordable, so you can use Script Editor, or Script Debugger, or any AppleScript tool that supports recording to help you get an idea of how to build simple AppleScripts. Even cooler are the new Action Menus, which live in the Finder, and are context sensitive based on the current selection. They kind of duplicate context menus, but not perfectly. Folder Actions get *much* easier to implement, thanks to context menu support in the Panther Finder. Select a folder, ctrl – click on

MAC OS® X PANTHER™

ALL YOU NEED TO KNOW ABOUT PANTHER

"The undisputed bible for advanced OS X issues"
—Mac OS X Guide



Mac OS X Panther Unleashed

by John Ray and William Ray

ISBN: 0-672-32604-3
$49.99 US

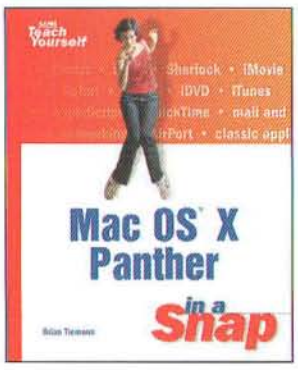
Underneath the colorful interface of Mac OS X is a powerful, complicated operating system based on BSD Unix. The third edition of Mac OS X Unleashed takes the same approach as the best selling first and second editions. Not only does this book help deal with the most trouble-prone aspects of the user interface, but also focusing to a great extent on the BSD environment and how the user and administrator can get the most out of the operating system.

VISIT **WWW.SAMSPUBLISHING.COM/MAC** FOR MORE SAMS MAC BOOKS

Sams Teach Yourself Mac OS X Panther All in One

by Robyn Ness and John Ray

ISBN: 0-672-32603-5 • $29.99 US

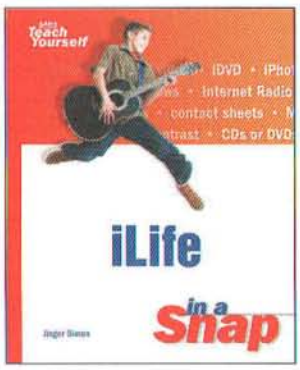


Mac OS X Panther in a Snap

by Brian Tiemann

ISBN: 0-672-32612-4 • $24.99 US

Available Feb. 2004



iLife in a Snap

by Jinger Simon

ISBN: 0-672-32577-2 • $24.99

Available Feb. 2004

amazon.com SAMS | www.samspublishing.com Available @ DEPOT

Amazon.com is the registered trademark of Amazon.com, Inc.

it, and you can attach, disable or configure Folder Actions. For me, this has been a real boon over the Jaguar method that required you to do it the hard way, or via a not terribly handy setup that required you to activate the Scripts menu in the Menu Bar. I have a "folder scan with Virex" folder action that I use to automatically scan files added to designated download folders, like my Desktop, the Entourage Saved Attachment folder, etc. I have another one that I use in conjunction with Distiller Watched folders that starts Distiller when something is added to a watched folder. A relatively minor feature that has made many scripter's lives easier.

NETWORKING

Since I am a network geek by trade, it behooves me to talk about Panther's networking changes. There have been some big changes here, some good, some bad, some just necessary.

Security

Panther is all about security. Almost everything in Panther has improved security. AFP and FTP services are now Kerberized if used with Mac OS X Server, so file transfers have gotten more secure. Even Mail has improvements to its Kerberos implementations, so for those of you using Kerberized email, (mostly Universities), Mail is a much nicer choice in Panther than it was in Jaguar. If your machine is managed by Mac OS X Server, you use the AD plugin for your logins, or you've customized your authentication setup, then you can get Kerberos tickets when you log in, which makes single signon much easier in Panther. (The idea with single signon, is that when you log into the machine, that's the only time you need to authenticate. After that, any resources you have access to, like network drives, email, etc. use various behind the scenes mechanisms, in this case Kerberos, to handle that, so you aren't having to deal with a daily stream of password requests.) In Panther, Kerberos is everywhere. Note that if you use AFS, or the Andrew File System as a distributed file system, you'll need to update to the most current version for Panther, from http://www.openafs.org/.

SSL gets a major boost in Panther as well. There is much better SSL support in Mail. application for using certificates with email, and S/MIME support for attachments. Safari's SSL support is a little better than it was in Jaguar, but still not as good as Mozilla's. Even relatively minor things, like the wake from sleep password dialogs are Kerberized. Apple has really done a good job of integrating security into Panther without making it a stumbling block to everyday 'normal' use. This kind of thing is not easy to pull off, but it's the best way to get everyone to start making secure computing something besides a buzzword at a trade show. The Keychain supports SSL better than it ever did before, which makes adding certificates to your system a much nicer process. Now, if the KeyChain scriptability were to be similarly improved, it would even nicer.

The password system in Panther has changed as well. New user accounts in Panther no longer use the old NetInfo 'crypt' passwords. Those were never really that secure anyway, and by getting rid of them, we get improved security, and much longer passwords. That's right, you can now have a password that's up to 255 characters in length. Not a bad improvement. Now, as to how you remember that, I have no idea. NetInfo is gradually going away anyway. LDAP is now the preferred protocol for Directory Services in Panther, and NetInfo is now only really needed for local machine records. I imagine that we'll probably see the end of NetInfo in the next few releases.

Windows connectivity

There have been improvements to non-Active Directory Windows networking. Samba 3 is now the back end for Windows connectivity in Panther. This allows for things like making your Mac OS X system an NT 4 Primary Domain Controller. (This is a simple UI setting in Mac OS X Server.) Winbind is now working in Mac OS X, so integrating Samba and Active Directory is a LOT easier under Panther than it was under Jaguar. If you have your Active Directory connections set up correctly, then single signon works pretty well for Windows file shares. (SMB printing is better as well, but printing gets its own section a little later on.) I haven't seen any real speed improvements as far as File Transfers go, and Apple *still* hasn't figured out how to not leave .DS_Store and other file boogers all over Windows drives. Thursby figured this out years ago, so Apple really needs to clean this up.

One trick that is totally new for Panther, and not really well known is that you can now access locally connected NTFS disks with Mac OS X. Apple incorporated the mount_ntfs utility that was a part of BSD into Panther. It's not a full implementation, (writing to NTFS drives is rather limited, and I'd not try it on a drive I cared a lot about), but I can see this being a last ditch way to get data off a Windows boot drive that won't mount right under Windows. Use 'man_ntfs' fro the full details.

Unix connectivity

This hasn't changed much at the file sharing level from Jaguar. You still can use NFS with some command line knowledge, or NFSManager, from Marcel Bresink, http://www.bresink.com/osx/NFSManager.html, which is still probably the best way to set up NFS on Mac OS X, regardless of version. Printing is still the same. It's LP(R), there's not much that can change there. The only big change is the integration of X11 into Panther, which allows you to access Unix applications and other non file and print resources, such as applications, etc. X11 also makes it easier to use things like MatLab, Open Office, and other local Unix applications that don't have, nor may ever have an Aqua interface. This is not saying that things like Fink are obsolete, but that you don't need to do as much work to get basic X11 functionality on your system.

Mac connectivity

With the major exception of having to manually enable AppleTalk browsing in Directory Access, the big changes involve Rendezvous, or Zeroconf. It's everywhere. File sharing access, printer discovery, Safari uses it, (Which is really cool for Rendezvous printers that have Web page configuration abilities), almost any file sharing you do from

W WINDOWS | M MACINTOSH | X MAC OS X

Thinkfree

ThinkfreeOffice

COMPATIBLE WITH MICROSOFT WORD, EXCEL AND POWERPOINT

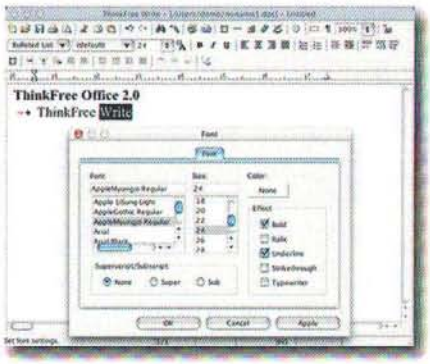
WORDPROCESSOR • SPREADSHEET • PRESENTATION GRAPHICS

The Affordable Office Alternative!

Three High-Performance Applications

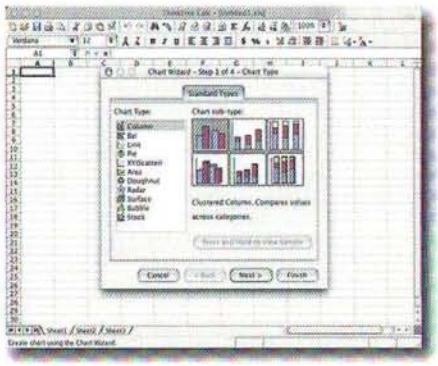
Thinkfree Write

Thinkfree Write is a powerful word processing application that enables you to create rich, professional quality documents and Web pages. You can insert tables, images, and clipart, or even apply custom layouts to your document...then effortlessly proofread your work with the easy-to-use spelling and auto-correction features.



Thinkfree Calc

Thinkfree Calc is a full-featured, easy-to-use spreadsheet application that can easily tackle the most complex analytical tasks with over 40 charts and 300 function capabilities. Thinkfree Calc opens, edits, and saves directly into the Microsoft Excel (.xls) format, so users can seamlessly share documents and collaborate with Microsoft Office users.



Thinkfree Show

Thinkfree Show enables you to create high-impact presentations including animation effects, drawings, images, clipart, and other graphic features. Thinkfree Show opens, edits and saves directly into the Microsoft PowerPoint (.ppt) format.

CyberdrivePlus

A free, one-year subscription to CyberdrivePlus is also included. CyberdrivePlus provides you with secure, Internet file storage and free online software upgrades!

ONLY $49.95

Macworld

"Thinkfree is a best-of-breed program that will exceed your expectations."
— *Jeffery Battersby*

MacAddict RATED GREAT

"Thinkfree Office is the next best thing and then some."
— *Deborah Shadovitz*

macHOME

"Thinkfree Office is an impressive attempt to crack the seemingly impenetrable productivity market."
— *Chris Ward*

amazon.com.
Apple Store
Apple Specialist
The Macintosh Superstore 1-800-873-3726
MacMall
Small Dog Electronics
Fry's ELECTRONICS
ZONES

Thinkfree and Thinkfree Office are trademarks of Thinkfree Corporation. All other trademarks and/or registered trademarks are properties of their respective owners.

a Mac running Panther is advertised via Rendezvous, so on a local subnet, finding another Mac's resources is dead simple. It's even a part of Terminal. That's right, Terminal is now Rendezvous – enabled. There's a new "Connect To Server" feature in Terminal, that when activated brings up a window with various connections and all the Macs it can find on the local link.

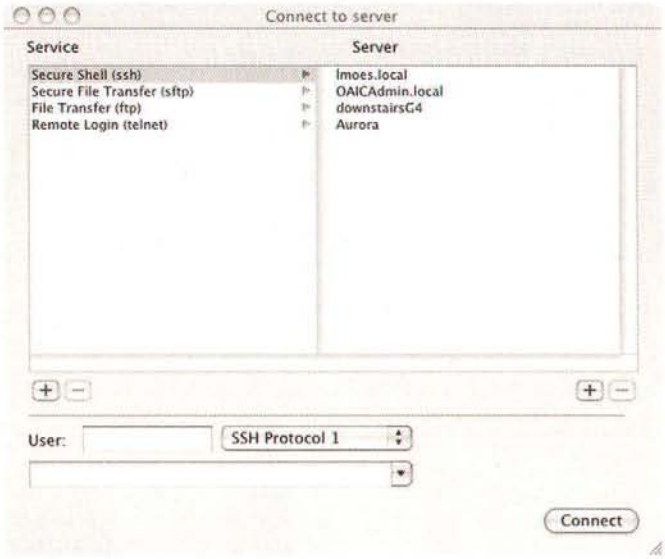


Figure 11. *Rendezvous and Terminal*

So, if you aren't sure as to the name of a Mac you want to connect to via SSH, SFTP, etc, just browse for it. If it's running Jaguar or Panther, you can find it. Very cool, and very unexpected. Leave it to Apple to make the Command Line simpler and easier to use. Speaking of Terminal, there are a few changes there, mainly a change to the default shell, (bash instead of tcsh), and the default terminal type, (x-term instead of vt100 or ANSI). These are more for compatibility with shell scripting needs and some other items, and other than a few potentially annoying behavioral changes, you shouldn't see any major problems coming from this. In any event, changing it back is no harder than changing your shell or terminal type was in Jaguar.

Browsing

There have been some major changes to how you browse networks in the Finder, and depending on your needs, this is either very good, or quite annoying. In Jaguar, all your network browsing was done via "Connect To Server" from the "Go" menu in the Finder. While not as disconnected as the Chooser, evidently it wasn't integrated enough for Apple. So now, if you want to browse for a network, you go to "Network" in the Finder, and start from there.

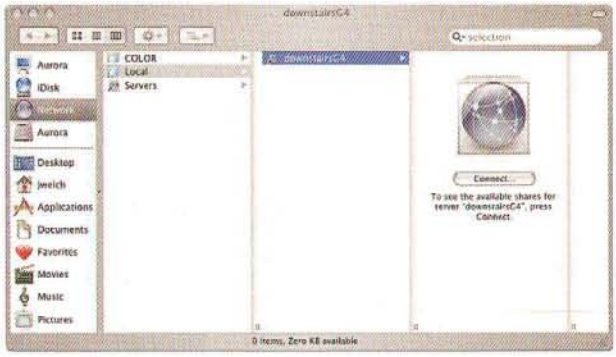


Figure 12. *New Browsing in Panther*

Now, you can see some obvious differences. First, this is done in the Finder directly. So you don't need Connect To Server anymore. However, for this to work, you have to be able to see the server you want via browsing. Secondly, with Connect To Server, you talk to the server, but mount the share. So even if a server had fifty shares, you are only mounting the shares you explicitly choose to mount. With Panther, and Network View, once you log into the server, you have access to every share on that server you're authorized for. So in effect, you are mounting the server, not the share. Another difference is where this mount connects to locally. With Connect To Server, the mounts showed up in /Volumes, just like any other drive. With Network View, the mounts live in /var/automount/Network. So if you rely on the path to a network share in a script, you may have to change some things. Shares on a server mounted via Network View don't show up on your Desktop. They only show up in Network in the Finder. So navigating to them can be a little tedious. Finally, you can't just unmount a share. You either unmount the server, and thereby unmount all the shares from that server, or you unmount nothing.

Connect to Server is still there, but severely limited compared to Jaguar. You cannot browse within Connect to Server, and you have to manually enter the URL for the server, or have the server in your favorites list. Any server mounted this way looks like it did in Jaguar. The mount lives in /Volumes, and it appears on your desktop. One thing to watch out for here is double mounting. If you connect to a server via Network View, then use Connect To Server with the same server, you can mount a share, or shares twice. This can cause you problems if you aren't careful.

Figure 13. *New Connect To Server Dialog*

Figure 14. *Oops, double-mounted share*

As you can see, the confusion potential is high. Especially with AppleScript, which could get really confused if the path to a share changes and a script is making now-incorrect assumptions. As well, the "mount volume" AppleScript command creates Connect to Server mounts, not Network View mounts.

Finally, since you can't obviously tell that you have fifty shares mounted, if you change network settings, or put the machine to sleep, and wake it up without a network connection, you're going to get a lot of "Oh dear, you appear to have disconnected from this share. Do you really want to do that? Are you sure" dialogs. It's an interesting change, but there are some real problems with it. Oh, one thing that seems to be fixed in Panther is the DNS serialization that made for some frustrating times in OS X prior to Panther. So, if the application is written correctly, one bad DNS lookup should not delay every other DNS lookup anymore.

Printing

Printing is of course, central to the Macintosh, and has been for many years. Now, I am not going to comment on the quality of fonts, etc, because to an IT geek like me, once you get beyond the network protocol, printing is magic. You tell the computer to print, and trees die. But there are some other changes to printing in Panther that even I can see. The most obvious one is the changing of the main printer configuration application from Print Center to Print Setup Utility. So far, the only problem the name change seemed to cause is with the Adobe Acrobat PDFMaker macro for Microsoft Office. Seems they hard-coded "Print Center" into the macro, since every time I've run it on Panther, it keeps asking me for the location of "Print Center". Other than that, I haven't seen anything break. (Well, it could make Quark unhappy, but determining that would require me to use Quark, and even I have limits to the amount of computer – induced pain I will voluntarily subject myself to.)

The basic UI in Printer Setup Utility is the same as it was in Jaguar. Once you get into the UI a bit, there are some very nice additions. You can now have Desktop or Dock printers. Just drag a printer to the Dock or the Desktop, (or wherever you want them), and you can now drag and drop documents on it. The UI for adding printers has gained a few tricks as well. For one, you can now add IPP and SMB printers without needing to hold down the option key when you click on the Add Printer button. When you add a printer, Printer Setup Utility does a much better job of discovering new drivers on your system. IP Printing now includes Rendezvous printers, LPR printers, IPP, and HP Jet Direct printers. If you have printers defined in an Mac OS X Server – hosted Open Directory domain, those show up in the Open Directory setting. Rendezvous gets its own heading in addition to being a part of IP Printing. (I'm also noticing that HP at least is behind Rendezvous in a big way. They have three models of laser printer that use Rendezvous by default, which makes for much easier network printing setup.) USB printing is the same, and Windows printing now has its own list item. This unfortunately doesn't make finding an SMB printer any easier, but at least you don't have to go through as many contortions.

The advanced selection option is still available, and if you have the Mac OS X native version of Distiller installed, you can set up a printer for that here, along with Fax printers, etc. Obviously some of the options available here are dependent on your hardware and software setup.

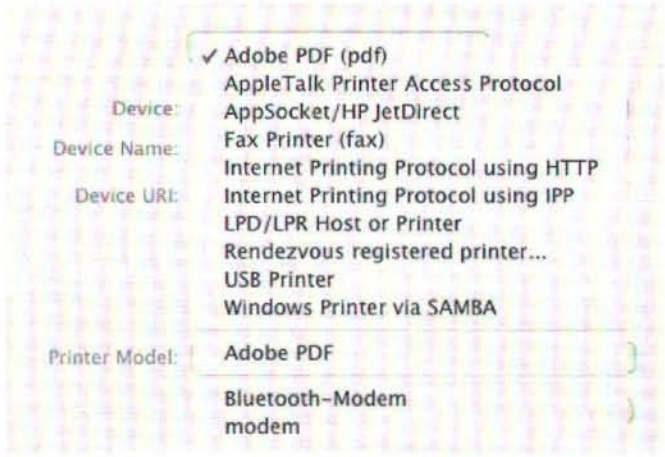


Figure 15. *Advanced Printer Setup Options*

Apple has also updated the printing event mechanism, so that if a developer takes advantage of it, you can automate printing to a much higher degree than was available in Jaguar, including bypassing the print dialog altogether. Obviously, this isn't just going to appear overnight, but the plumbing is now there. While we're talking about automation, Panther still hasn't fixed one major remaining problem, the fact that printer creation is still essentially a manual process. You can't use AppleScript to create printers. You can use it to see status on printers and jobs, and set the default printer, but even simple items like printer status are read only, so you can't easily control printing via AppleScript. You can do this all via shell scripting, and do shell script, but AppleScript print control is a critical part of the printing workflow, and Apple still not making AppleScript and

print control/creation usable is definitely frustrating, and starting to border on unacceptable. If I have to use shell to automate printer control, then it makes more sense from a fiscal point of view to use Linux, or some other Unix box as a print server, and save some money over the cost of a Mac. A dedicated print server gains nothing from a nice UI, and if Apple forces you to use standard Unix methodologies to control printing anyway, what's the point in paying the premium?

AppleScript

This brings us to the next major part of Mac OS X, and one of the major features of the platform as a whole, AppleScript. There have been a lot of improvements, but there are still some really glaring holes, although, thankfully, most of those exist outside of the OS. (I really don't see any of the Pro applications ever becoming scriptable, not a good example of eating one's dogfood.)

The version of AppleScript that ships with Panther is not the long – awaited AppleScript X. Instead, we get version 1.9.2 with Panther. This is not to say that there are no real improvements, not by a long shot. The Finder is partially recordable again, so newcomers to AppleScript can see how to create bad syntax automatically. (That's not a shot at Apple as much as an acknowledgement that an AppleScript version of manual actions is not going to be an example of efficient AppleScript code.) Nonetheless, it's about time that one of the first applications anyone wants to script is finally recordable.

In Panther, you can send Apple events not only to a remote machine, via the eppc:// url, but also to a specific user id, and process id. Because of Fast User Switching, you can't assume that any one particular user is going to be the active session. So, you can target the specific user id and process id on a remote machine. This has some nice potential for production systems, in that you can start up a machine under a production user, start various processes, the switch back to the login screen, so if humans need to use it, they can without logging out that user. So, if you know a process is running, you can remotely determine the pid, and then target that pid and that production user with a script, and not have to worry about what another user is doing. If you don't specify uid and pid, then the current active user is targeted. If you aren't root, then you can only target processes that match your uid. The eppc server is now a part of xinetd, and is advertised through Rendezvous.

There is a new script format as of Panther, the script bundle. This is an AppleScript version of an application bundle, and while only usable with AppleScript 1.9.2, you get some nice features with it, namely the ability to include Scripting Additions with the script application. One of the big problems in using Scripting Additions has been the fact that you couldn't ensure that everyone using your script would have the required additions. This is no longer a problem, as you can place the Addition in the bundle, and it will be used when the application is run. Apple events are now handled as FIFO instead of LIFO, important for CGI uses. The "path to" command has been enhanced, with new destinations, and the ability to create a folder if it doesn't exist. It also has better support for Classic. You can get the long and short names of a user far easier than in Jaguar.

Some bug fixes include the ability to deal with https:// urls, support for larger numbers, better handling of a "quit" event sent to an application that isn't running, (it no longer starts the application just so it can quit it), and do shell script now handles failed authentication differently than canceled authentication.

There are still some really annoying holes. Keychain Scripting hasn't been updated to allow you to specify which applications have access to a given key, although the protocol setting bug has been fixed. I've also run into a problem where Keychain Scripting can't access the Keychain I brought over from Jaguar, but it can access any keychains created under Panther. I can't create network configurations in panther, but I can script Internet Connect. So as long as all you need is what Internet Connect gives you, you're golden. Switching network configurations via AppleScript requires far too much work. QuickTime allows you to use AppleScript for everything but creating custom export parameters. Xcode can't debug AppleScripts beyond event logging and "display dialog". There's no find function in the Finder dictionary. There's no "as password" ability in display dialog, so if you use it to ask a user for a password, the password is displayed as clear text. It's too tricky to run scripts when the machine is shut down or restarted. The ability to run a script when the machine wakes up would be greatly appreciated by many. Too many OS utilities use non-standard UI widgets, so UI scripting doesn't work with them.

So AppleScript is, as always a mix of good and bad, of parts given proper attention, and others seemingly ignored by Apple entirely. For the AppleScript glitterati, this is just part and parcel of the interesting way Apple has treated AppleScript since its invention, and it guarantees that scripters never get bored.

Conclusion

Well, we've covered a lot, and there are volumes I haven't even touched on. But that's how it is in a review. I tried to get to areas I haven't seen covered in other reviews, and avoid the ones that have been done to death, (Expose!). From the networking/IT point of view, there are a lot of needed improvements. One thing that should also be mentioned, although I've never gotten why it's important as anything more than a trivia item is the boot and shutdown times. Both are *much* faster under Panther. My 17" PowerBook can go from a cold start to a login screen in well under a minute, and shutdown averages less than ten seconds, both a noticeable improvement over Jaguar, particularly the shutdown time.

From my POV, Panther contains enough improvements to justify the upgrade cost with ease, and if you have to play on non-Mac networks, Panther is head and shoulders above Jaguar. If you haven't made the jump yet, I hope this review was of some help to you.

32

< 1 year

2 years >

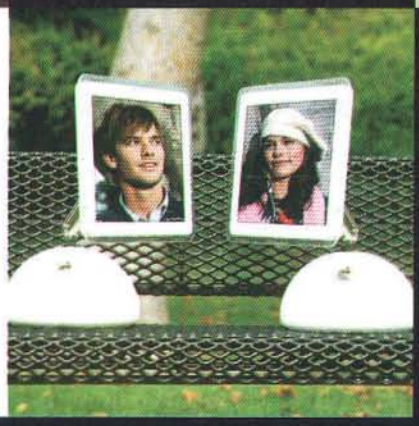
$64

Interviews

Tapping into the world of celebrities and their Macs, only MacDirectory offers exclusive interviews. Get a close and personal view from Sarah Jessica Parker, Sting, Steve Jobs, Madonna, Harry Connick Jr., George Lucas, Jennifer Jason Leigh, Steve Woz and other leaders in the Mac community.

Features

Designers, writers, musicians, business leaders & our technical expert team offer their own personal interpretation of things that only the Mac system can deliver. Featuring over 240 pages of reviews, interviews, news, insights, trends and the largest Macintosh buyers' guide including over 5,000 Mac products and services.



Culture

MacDirectory takes you to the wildest corners of the world and uncovers how Macintosh computers are being used by other cultures. Travel to Japan, Australia, Germany, Brazil, India, Russia & learn more about Apple's cultural impact around the globe.

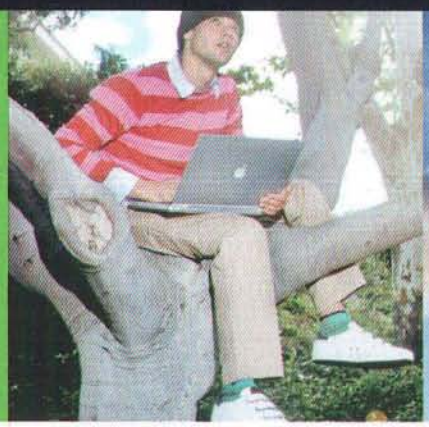
MacDirectory

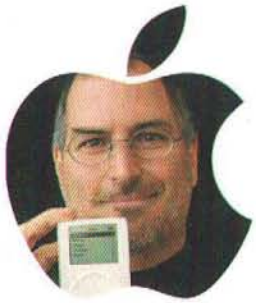
Beyond any Macintosh Magazine. Subscribe.

< Subscribe

ww.macdirectory.com/mw.html

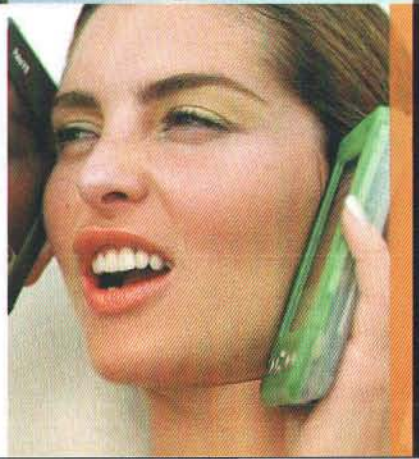
SEND CHECK OR MONEY ORDER TO:
MacDirectory Sub Dept.
326 A Street, 2C
South Boston, MA 02110





Reviews

Find out all you need to know about the latest Mac products including the hottest Mac OS software and hardware.



Win!

Subscribe to MacDirectory and you will automatically enter our sweepstakes for a chance to win a new Titanium!

By Schoun P. Regan

Taking Mac OS X Server for a ride

A brief look inside Panther, Apple's latest update to Mac OS X Server

It all began with Rhapsody. The green box with the gears, or, Mac OS X Server 1.0, as named by Apple. It had NetInfo at the core and support for LDAP. It supported Classic with OS version 8.6 and it introduced Macintosh users to the UNIX filing system.

A few years later, Apple introduced Mac OS X Server 10.1 which added a more refined GUI, enhanced administration tools, and withdrew, rightfully so, a default install of Classic.

Mac OS X Server 10.2, code named Jaguar, brought us Zero Conf in the form of a marketing term called Rendezvous, improved integration of Samba; the open source project responsible for sharing with Windows' computers, network installations, and Open Directory, the name Apple gave to Mac OS X's ability to talk to other directory services, such as LDAP, Active Directory, NetInfo, and NIS.

Enter Panther, Mac OS X Server 10.3. Panther changes the rules. Panther is the most open-sourced, well thought out, scalable software to come out of Apple's campus. But just like a car, reading the features list doesn't really give you a feel for the product. Let's take a look at how Mac OS X Server changes the rules, ups the ante, and stands up to the rest of the pack.

Taking the time to install the server the first time means answering questions about the network information of the server, the services, the directory setup, and a few more options. Reinstalling a server from scratch means that you must locate the piece of paper where you wrote down all the settings. If your office is as cluttered as mine, you'll never find it. Mac OS X Server now includes the ability to save the server settings setup, allowing you to instantly apply those settings to another server upon install. The property list configuration file can be encrypted and stuck on your iPod for safekeeping. This is an enormous advantage when setting up servers. A configuration file can be used to mirror a server setup across the country by emailing the configuration file to the remote server administrator. Better still; Mac OS X Server will search a network for this file when starting up, so starting over from scratch is not so difficult anymore.

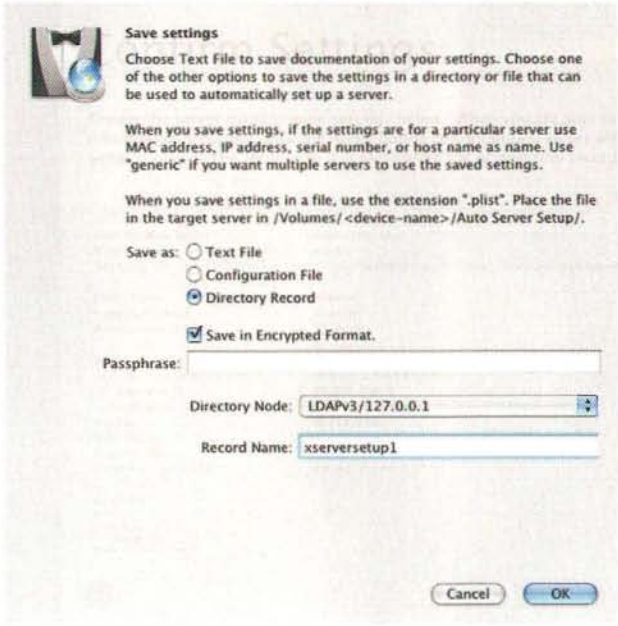


***Figure 1**: Dialog permitting the saving of Mac OS X Server settings.*

Once the server is up and running, three tools are utilized for the administration and monitoring of Mac OS X Server; Server Admin, Workgroup Manager, and Server Monitor. Server Admin is a conglomeration of the old Server Settings and Server Status tools, while Workgroup Manager and Server Monitor tasks remain unchanged.

Let's begin with Server Admin, which is used to configure and monitor services provided by Mac OS X Server. If you have a service you wish to add to Mac OS X Server, you now have the ability to manage that service from Server Admin by writing a plug-in for your service. For example, Sybase could write a plug-in for Server Admin, allowing the administration

Schoun P. Regan is CEO of The Mac Trainers, which specializes in Mac OS X training and consulting. He speaks regularly to CEOs and CFOs on how to control IT department spending, the myths surrounding cross-platform integration, and the lunacy of expected lost revenue stemming from a culture bred to tolerate IT staff, and operating system, inadequacies as "normal". He seeks to change self-fulfilling IT departments that breed complacency for their jobs and contempt for the end user, neither of which are conducive to business.

of that Sybase database from the Server Admin tool. Apple has provided a consistent and relatively easy to use interface in Server Admin. So, now any service can be carefully configured and tested. Their settings can saved to a small property list file by simply dragging a small icon off the service's window to the desktop. Again, we see that these small configuration files can be used to configure identical service settings on any other Mac OS X Server, or used to recover existing settings when additional configuration changes go terribly awry.

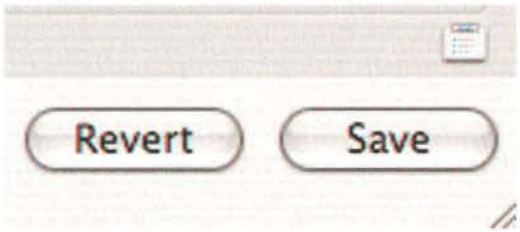


Figure 2: *Small configuration box above the Save button.*

Apple included BIND v9.2.2 with this version of Mac OS X Server and the interface for DNS now permits the creation of Zone files and their subsequent records. What is not automatic yet is the default creation of reverse lookup files and records. You must place a check box in the Address record setup indicating you want to have a reverse lookup file created. Since Mac OS X Server relies heavily on DNS, insuring your records are accurate is essential to keeping other dependant services running smoothly.

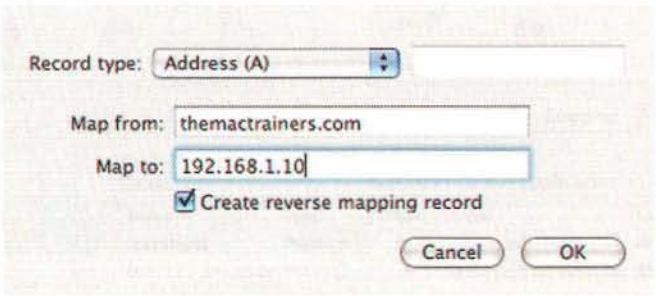


Figure 3: *Adding a DNS Address record.*

Two other GUI additions to Server Admin are NAT and VPN. Prior to this version of Mac OS X Server, if you wanted to do Network Address Translation, you needed to pop into the command line and configure your routing table. While the GUI for NAT is not particularly spectacular, you can choose the interface on which it runs, which is really all you need for basic NAT. The VPN piece includes the ability to setup both PPTP and L2TP over IPSec. Anyone with the authorization to tunnel into the network can now do so.

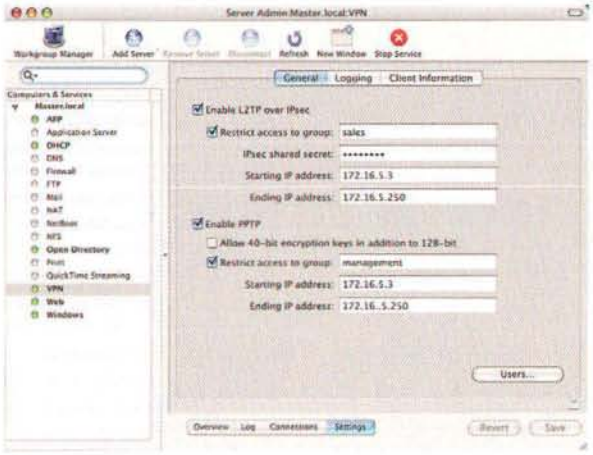


Figure 4: *IPsec and PPTP options*

Now, it's time to welcome back Open Directory, renamed called Open Directory2, Apple's marketing name for how Mac OS X integrates with other directory services, such as NIS, NetInfo, LDAP, and Active Directory. Apple has included an Active Directory plug-in that requires no schema modifications on the AD server, a Windows Active Directory administrator's dream. If the Active Directory schema has been changed, the plug-in module will access information within the modified schema, including any MCX settings that may have previously existed. If schema modifications have not been done, the plug-in will automatically generate the required attributes needed. Advanced options include caching user logon information for laptops, multiple domain authentication, and group administration settings. This feature is available on both Mac OS X and Mac OS X Server 10.3.

Figure 5: *The Active Directory PAM (plug-in Authentication Module) for Open Directory*

Another part of Open Directory's job is to assist the administrator when creating a secondary database on Mac OS X

A Universe of Solutions

Enhance your knowledge and skills by attending the world's most comprehensive forum for Mac users. Transcend your boundaries and make informed purchasing decisions.

January 5-9, 2004

San Francisco, CA
The Moscone Center

Conference	January 5-9, 2004
Expo	January 6-9, 2004

Macworld
Conference & Expo

Find out what all the buzz is about at
www.macworldexpo.com

Flagship Sponsors:
IDG WORLD EXPO Macworld

Macworld Conference & Expo,® 20 years young, will deliver you a universe of solutions for all your Macintosh needs. Don't miss your opportunity to experience everything Mac in one place!

- Acquire knowledge to stay competitive
- Test drive new and innovative products
- Network with like-minded people
- Talk with the companies that make you more productive
- Feel the power of this holistic community
- Enjoy the atmosphere

Register online with Priority Code: A-MTD

Macworld.com **MacCentral**

Server. When you first install Mac OS X Server, a local NetInfo database is created. This database is for local users, which are usually server administrators. Creating a Master database creates an LDAP database for networked users. Once a Master database is created, system administrators have the ability to configure various attributes via the built-in GUI, which also adds a new Inspector window (under Workgroup Manager Preferences). Once your Master is configured, add all the users, configure all their options, and then you can replicate that server to any other server, allowing both redundancy and speed of authentication since a network user does not have to cross subnets to authenticate against the Master if the Replica is nearby. So, for example, the Master server could reside in the school IT administrator's office downtown while one or two replicas exist in each school building. This permits students in one school building to authenticate locally, without sending the IP traffic downtown. This could be done in prior versions of Mac OS X Server with NetInfo. It was called cloning a database but there was no documentation for it. Master/Replica configurations rely on DNS and can be configured to update automatically as the Master changes or periodically, based on the Master settings.

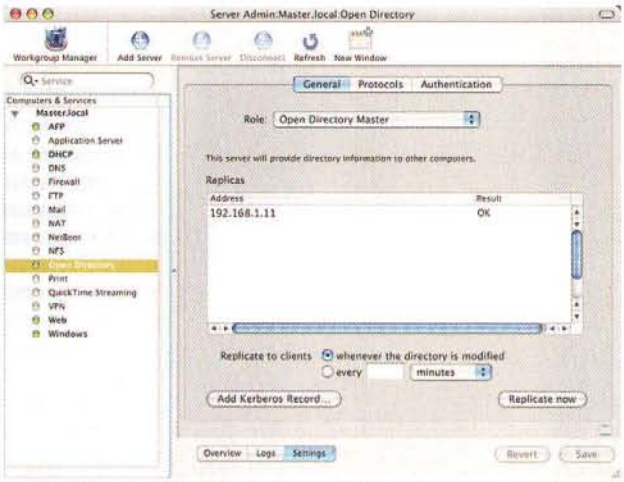


Figure 6: *Open Directory configuration as a Master.*

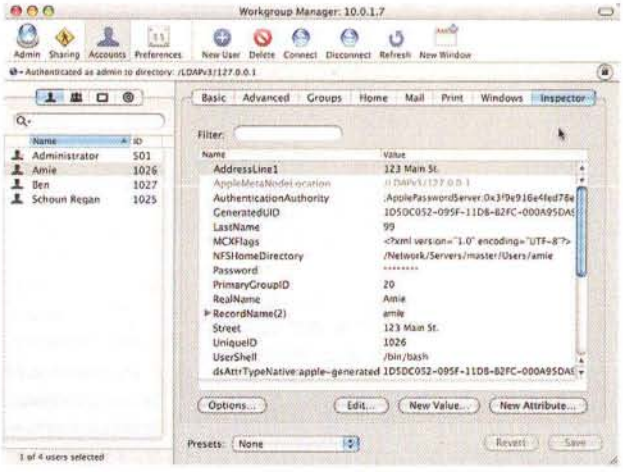


Figure 7: *The Inspector tab of Workgroup Manager*

Security rules in Mac OS X Server. Not only does Apple now create a Kerberos KDC every time you create a Master, but they have cleaned up some older, less secure authentication methods. Mac OS X Server can use Kerberos, Password Server, and ShadowHash passwords to authenticate users. It does support crypt passwords if you are importing users from previous NetInfo databases, but you cannot create new users and assign them a crypt password. Password limitations have also been altered to include a 511 byte length for Password Server. Bytes length limitations are used instead of character limitations in case UNICODE fonts are used. Suffice to say that the password can be longer that you ever thought possible for a typical user. Encryption methods supported include DHX, CRAM-MD5 and Digest-MD5, SMB-NT, SMB-LAN Manager, and MS-CHAPv2.

Mac OS X Server does Windows like before, but again, support is improved. In previous versions you've been able to let Windows users authenticate against a Mac OS X Server and share files across platforms, but that's where it ended in the GUI. Now you can add a Windows Primary Domain Controller to the mix. This helps you close the gap on that single sign-on utopia most IT managers yearn for.

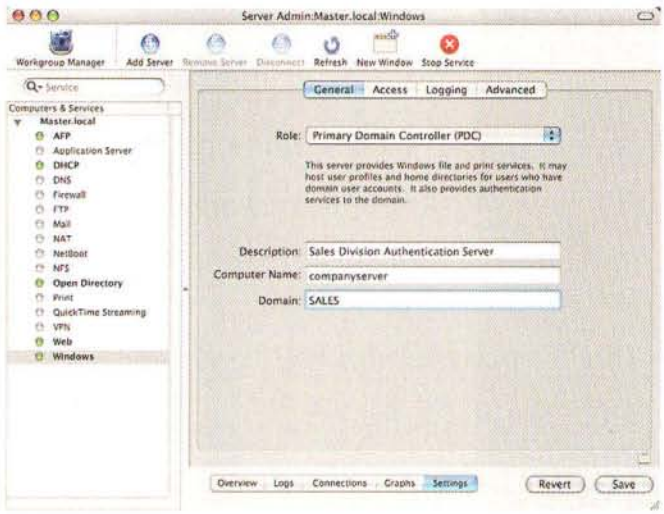


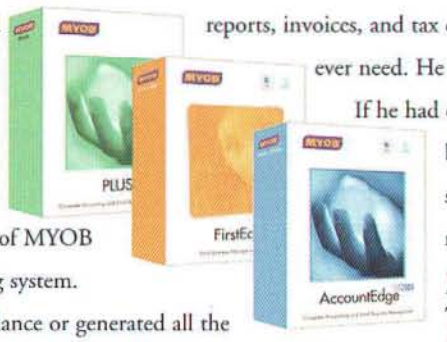
Figure 8: *Windows PDC Setup window*

There are other services worthy of mention, DHCP, NetBoot, Firewall, QuickTime Streaming Server, Mail, and more. But what stands out about the latest incarnation of Mac OS X Server boils down to one word: interoperability. Being able to use open standards, communicate with other systems, and scale well are all hallmarks of what we look for in robust, commercial, full-grade server systems. Apple has hit the mark this time, and has the server world square in its sights.

See Dick. See Dick run his business with software that wasn't written and designed for his Macintosh. Poor Dick.

A moment of silence for Dick, please. A good guy with a good small business, but his accounting software was one of those PC transcription jobs, not pure MAC like MYOB AccountEdge and MYOB FirstEdge.

If only he'd known about the amazing capacity of MYOB software to bring out the best in his MAC operating system. He could have tracked and managed finances at a glance or generated all the reports, invoices, and tax documents that he and his accountant would ever need. He could have spent more time with his clients.



If he had only known that MYOB develops the world's best selling MAC small business management software for lots of good reasons, this story might have had a happy ending. Sorry Dick.

MYOB,
THE MAC ANSWER.

©2003 MYOB US, Inc. (800)322-MYOB www.myob.com/us

By Richard Gaskin

In Praise of 4GLs

Making a custom Internet application in less than a day

Introduction

Fourth-generation languages (4GLs) have grown up. Once limited to narrow tasks like database queries, modern 4Gls, like Revolution and SuperCard, offer rich languages and object models suitable for complete GUI application development.

Using a 4GL can be a smart choice when developer productivity is a project's critical driver. For example, many vertical market applications are too specialized to have a broad enough audience to support development in lower-level languages like C++ or Java, but a 4GL can often get the job done in a fraction of the time.

The tradeoff for this ease of development is sometimes execution speed, as the interpretation of scripts at runtime involve steps done at compile-time in lower-level languages. However, with well-optimized 4Gls, like Revolution, this is not always the case, and for many common tasks Revolution measurably outperforms Java, and other lower-level languages.

To better appreciate the runtime efficiency of a modern 4GL like Revolution, consider what's happening under the hood. In Revolution, when a script is loaded, it's interpreted into bytecode in a form reported to be more efficient than Java's. When executing a script, this bytecode acts as a glue, connecting compiled routines in the engine written in C++. So, while a script is indeed interpreted, a single line of script often triggers the execution of hundreds of lines of compiled, optimized C++.

To illustrate the productivity gain, consider the task of creating an alias to a file. In C it takes about 18 lines, and in a Java example posted to an Apple discussion list it took 247 lines. In Revolution's scripting language, Transcript, it's a one-liner:

```
create alias MyAliasPath to file MySourcePath
```

With hundreds of commands, functions, and object properties in Transcript, things like opening windows, handling controls, reading files, and other common tasks are usually one-liners, which means that most of the code being executed is natively compiled C++. In essence, the Transcript engine acts like a precompiled object library you string together with a few lines of script.

Productivity in Action

Let's take a look at how these productivity benefits play out in a real-word example: building a custom FTP client in a few hours.

The need for this arose from a teleconference with a client. We were starting a project in which we'd need to trade a lot of files back and forth. The number of files made email a cumbersome option, and the size of some of them made using email prohibitive.

Once we decided to use FTP we reviewed the various FTP clients available, including my personal favorite, Interarchy. While the applications we looked at were all great general solutions for file transfer, the breadth of features they offered was problematic in our case, given that files would sometimes need to be transferred by people with little or no experience using FTP tools. Also, the client's office had a mix of machines running OS X and Windows, so we wanted something that was not merely easy to use, but also had the same simple interface for both platforms.

I offered to make a custom FTP tool with the goal of having the fewest features possible. It would be hard-wired to work with a single directory on our server, require nothing more than a password to log in, and work identically on both OS X and XP.

When I told him I'd have it ready by the end of the day he was surprised. What he didn't know was that the secret weapon making this possible was Revolution's Internet library, libURL.

libURL: The Heart of our Application

Tim Monroe's excellent series of MacTech articles on developing QuickTime applications with Revolution provides a great introduction to the development environment and scripting language, so here we'll focus on working with libURL.

Richard Gaskin is president of Fourth World Media Corporation, a Los Angeles-based consultancy specializing in multi-platform software development. With 15 years' experience, Richard has delivered dozens of applications for small businesses and Fortune 500 companies on Mac OS, Windows, UNIX, and the World Wide Web. http://www.fourthworld.com

The Revolution engine contains robust support for TCP and UDP sockets, but if you've ever written an FTP client, you know how difficult it can be to handle all the error-checking needed to do it right. So, along with the other scripted libraries included with Revolution is libURL, which provides a simple interface for handling HTTP and FTP transactions.

LibURL uses Transcript's simple **put** command for the most common tasks. Just as you would display text in a field with **put "Hello World" into field 1**, you can specify a URL as a container as well.

To download a file from a Web server and display its contents in a field:

```
put url "http://www.fourthworld.com/data.txt" into field 1
```

Uploading a file to an FTP server is just as easy:

```
put tMyData into url \
  "ftp://user:password@ftp.domain.com/tDataFile.dat"
```

LibUrl also provides more than a dozen other commands to allow asynchronous transfer, logging, status updates, and more.

With libURL and Revolution's simple tools for building interfaces, putting the application together was a snap.

Development Chronology

8:30 AM Defined the requirements with the client.

9:00 AM The simple interface needed only a few objects, including a list field to display remote files, a **Download** button, **Upload** button, and of course a button to initiate the connection. **Figure 1** shows Revolution's Tool palette for creating objects, and its Inspector palette for setting object properties.

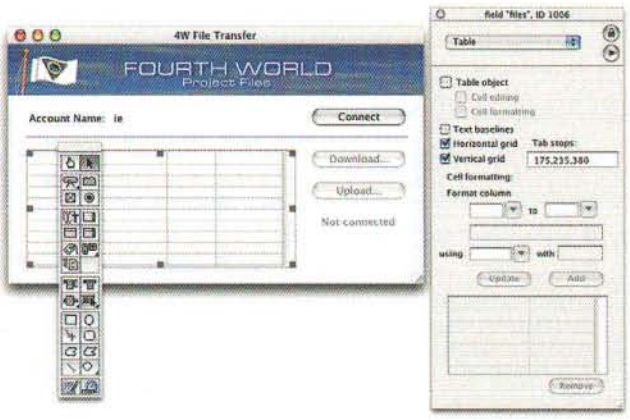


Figure 1. *Building the Interface*

Figure 2 shows a profile of our objects in Revolution's Application Browser after adding a menu bar and a hidden progress bar which we'll show during file transfers.

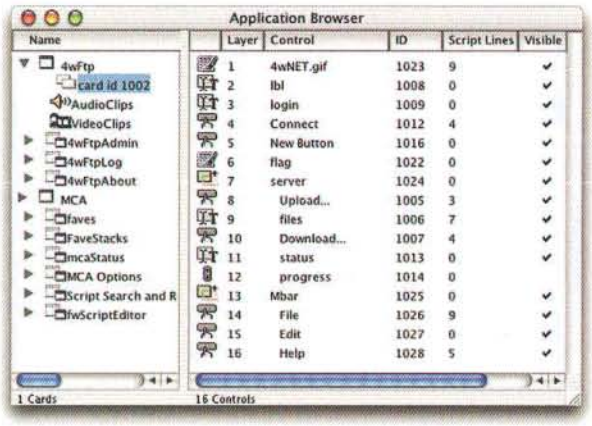


Figure 2. *Layout Overview*

Once the client interface was done, I needed to build an administration window for my own use during development, so I could enter the FTP login information. I could have written those settings directly into the code, but taking a moment to make an interface for myself will make it easier to modify the application for other projects in the future (**Figure 3**).

Figure 3. *Admin window*

I also added a simple window containing only a field to display a log of the transactions with the server to help with debugging. This exploited one of the conveniences of libURL: it includes a simple command that lets you use any field object as a log. You just pass it any valid field reference and all TCP transaction logging is automatically appended to the field's contents as it goes:

```
libUrlSetLogField the long ID of fld "log" \
  of stack "4wFtpLog"
```

The graphics in the main window were just modified versions of images from our Web site, so the interface was completed in about an hour.

10:00 AM With the interface objects in place it was time to start coding. The first thing the program needed to do, of course, was display a list of files from the remote server. As shown above we can download a file by just using the get url command, and we can download a directory listing by just specifying any valid directory on the server with the trailing slash:

```
get url "ftp://user:password@ftp.domain.com/directory/"
```

This returns a raw listing of files in a standard directory format:

```
drwxr-xr-x   2 user2000 user2000       512 Oct  1 01:46 .
drwxr-xr-x   4 user2000 user2000       512 Sep 30 08:44 ..
-rw-r--r--   1 user2000 user2000    898089 Oct  1 01:48
Tutorial1.pdf
-rw-r--r--   1 user2000 user2000   1118961 Oct  1 01:46
Tutorial2.pdf
-rw-r--r--   1 user2000 user2000    557660 Oct  1 01:44
Tutorial3.pdf
-rw-r--r--   1 user2000 user2000     88486 Oct  1 01:41
workflow2.png
```

If order to display the file list in a simpler form for the user, the first function I wrote converted the raw data into a tab-delimited list, since Revolution fields can display tab-delimited data in a field object as a multi-column list automatically.

Converting this raw data introduced one of the strengths unique to Revolution and other 4GLs inspired by HyperTalk, affectionately called *chunk expressions*. These HyperTalk dialects support fast and simple text parsing with references like word (space-delimited), item (comma-delimited), and line (return-delimited). In Transcript all three of these chunk types can use custom delimiters so they can be applied to a wide variety of tasks. Similar routines in other languages usually require walking through blocks of text one character at a time, counting delimiters, and building arrays as you go. In Transcript, however, you can simply write things like:

```
get word 2 of item 3 of line 4
```

Here's the code for the reformatting function:

```
-
- FormatRemoteFileList
-
- Extracts the relevant into from each line in the
- FTP file list passed in pList and returns a tab-
- delimited list of just file name, size, and date
- for display in the file list control
-
function FormatRemoteFileList pList
   put empty into tFormattedList
   repeat for each line tFile in pList
      -- Skip folders in this version:
      if char 1 of tFile = "d" then next repeat
      -- Get file name:
      put word 9 to (the number of words of tFile) \
        of tFile into tName
      -- Skip invisible files:
      if char 1 of tName = "." then next repeat
      --
      put Bytes2Size(word 5 of tFile) into tSize
      put word 6 to 8 of tFile into tDate
      put tName &tab& tSize &tab & tDate \
         &cr after tFormattedList
   end repeat
   delete last char of tFormattedList
   return tFormattedList
end FormatRemoteFileList
```

The Bytes2Size function simply converts bytes into a common abbreviation appropriate for user display, for example "323455" is returned as "3.2Mb":

```
function Bytes2Size n, pPadFlag
   if pPadFlag is empty then set the numberformat to "0.#"
   else set the numberformat to "0.0"
   --
   if n < 1024 then put n &" bytes" into n
   else
      put n / 1024 into n
      if n < 1024 then put n &"k" into n
      else
         put n / 1024 &"Mb" into n
      end if
   end if
   return n
end Bytes2Size
```

Since most of the controls would affect the user interface, I wrote one handler to handle all interface updates, called from most of the other UI-related code. I wanted to keep things simple, so I had it pass just one parameter to indicate if the user was connected to the server so it would refresh the file list. If called with no arguments, it cleared the list field:

```
-
- UpdateConnectionUI
-
- One-stop shopping for updating the user interface
- whenever the user connects or disconnects
-
on UpdateConnectionUI pConnectedFlag
   libUrlSetStatusCallback
   put (pConnectedFlag = "connected") into tIsConnected
   set the enabled of grp "server" to tIsConnected
   hide scrollbar "progress"
   if tIsConnected is true then
      set the label of btn "Connect" to "Disconnect"
      --
      put ServerDirectoryUrl() into tUrl
      put url tUrl into tFileList
      if (the result is not empty) and \
       ("not completed" is not in the result) then
         answer the result
         UpdateConnectionUI
         exit to top
      end if
      put FormatRemoteFileList(tFileList) into fld "files"
      PutStatus "Connected"
      --
   else
      set the label of btn "Connect" to empty
      put empty into fld "files"
      PutStatus "Not connected"
   end if
   disable btn "Download..."
end UpdateConnectionUI
```

That handler called a couple of other simple handlers written as a convenience:

```
-
- ServerDirectoryUrl
-
- Returns the full URL to the server directory
- by concatenating info stored in the hidden
- Admin window
-
```

Technological.

The logical way to connect your news and information to the people who count.

At PR Newswire, our approach to distributing technology news is simple: target the people with a real interest in your news – journalists, investors, analysts and consumers – and deliver your message to them in the way they prefer.

PR Newswire has earned the respect of the media and financial community by consistently delivering accurate, credible, reliable news and information directly to their desktops. We also offer a full range of customized solutions to help get your message across with photos, VNRs, Webcasting, and more.

Make sure your news is seen in all the right places.
Call PR Newswire at 888-776-0942 or visit us at www.prnewswire.com.

We tell your story to the world.™

PR Newswire
United Business Media

© PR Newswire Association LLC 2003

```
function  ServerDirectoryUrl
   return "ftp://"& \
         fld "login"     of stack "4wFtpAdmin" &":"& \
         fld "password"  of stack "4wFtpAdmin" &"@"& \
         fld "server"    of stack "4wFtpAdmin" &"/"& \
         fld "directory" of stack "4wFtpAdmin" &"/"
end ServerDirectoryUrl

--
-- PutStatus
--
-- Displays the string in s in the "Status" field
--
on PutStatus s
   put s into field "status" of stack "4wFtp"
end PutStatus
```

With the basics in place it was time to put them to work by adding code to handle the **Connect** button. To keep the interface as simple as possible, I used one button object for both connecting and disconnecting, merely changing the button's label property to reflect the change to the state. This is from the **Connect** button's script:

```
on mouseUp
   if the label of me = "Disconnect" then
     DisconnectFromServer
   else ConnectToServer
end mouseUp
```

ConnectToServer and DisconnectFromServer are defined in the script of the main window, which Revolution refers to as a "stack":

```
--
-- ConnectToServer
--
-- Called from the Connect/Disconnect button
-- to log on to the FTP server and obtain a
-- list of files
--
on ConnectToServer
   put fld "Login" into tLogin
   if tLogin <> fld "login" of stack "4wFtpAdmin" then
      answer "Incorrect login"
      exit to top
   end if
   --
   set cursor to watch
   put fld "login" into tLogin
   ask password clear "Enter your password:" as sheet
   if it is empty then exit to top
   put it into tPassword
   if tPassword <> fld "password" of stack \
    "4wFtpAdmin" then
      answer "Wrong password for your site."
      exit to top
   end if
   --
   if "4wFtpLog" is in the windows then
      libUrlSetLogField the long ID of \
       fld "log" of stack "4wFtpLog"
   end if
   --
   UpdateConnectionUI "connected"
end ConnectToServer

--
-- DisconnectFromServer
--
-- Called from the Connect/Disconnect button to
-- clear the file list and update the interface
--
on DisconnectFromServer
   UpdateConnectionUI
end DisconnectFromServer
```

Most of the code is fairly self-explanatory with the background provided earlier, but it's worth calling your attention to the ask and answer commands. These Transcript commands provide one-line convenience for displaying a simple alert dialog with answer, or allowing user input with ask, returning values in a predefined local variable named it. Extensions to these commands also provide a script interface to the OS's GetFile and PutFile dialogs on each of the supported platforms, which will be used later when we script the **Download** and **Upload** buttons.

I tested the work done thus far, clicking the **Connect** button to see the remote file list displayed and the button relabeled to **Disconnect**. I clicked it again to clear the interface. So far, so good. Time for lunch.

12 Noon With things going so well I packed a picnic and took it to the deck on the roof of my office building to enjoy a long lunch in the California sunshine.

1:30 PM Returning to the office, the next step was to get the Upload and Download buttons working. Both of these make extensive use of a great feature of libURL, the libUrlSetStatusCallback command.

As we covered earlier, Transcript's get and put commands can be used to conveniently use URLs as containers. While these are very convenient, they are synchronous, suspending other script execution such as one might need for updating a progress bar, for example.

Fortunately, libURL provides commands for asynchronous data transfer, as well. For FTP, these commands include libUrlDownloadToFile and libUrlFtpUploadFile.

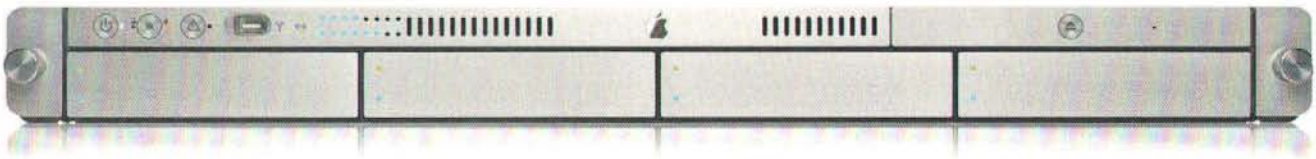
But, to get the most out of asynchronous execution, you'll want to be notified of the status of the transaction so you can take appropriate steps, such as updating a progress bar, and notifying the user when the transfer is completed. You set this up with libURL using the libUrlSetStatusCallback command, specifying the message name you want sent and the object it should be sent to:

```
libUrlSetStatusCallback message, object
```

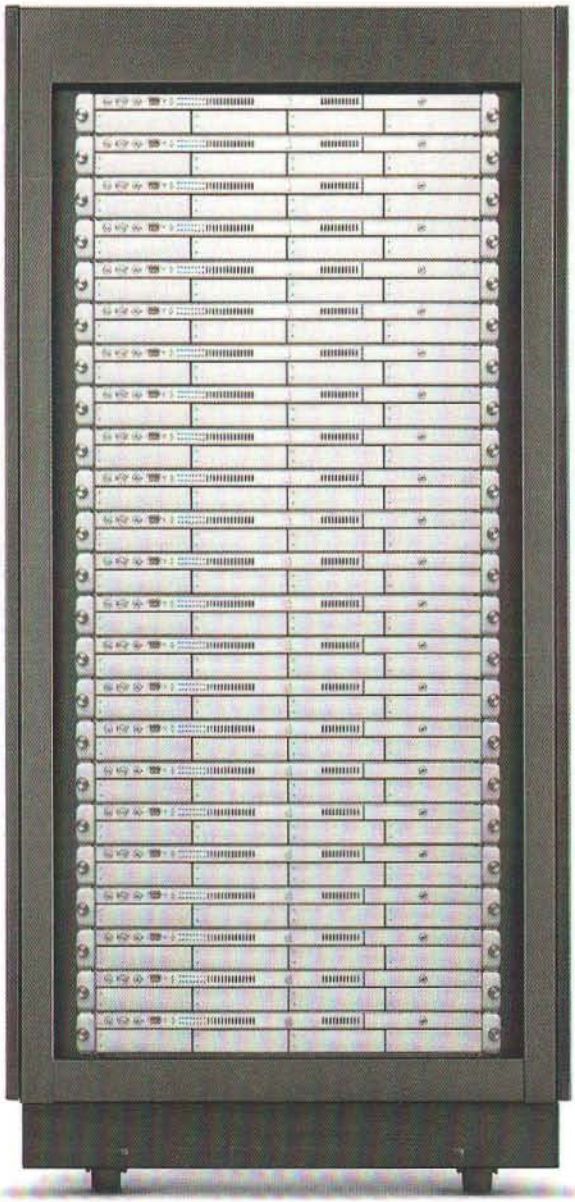
While a transfer is in progress, libURL will then send that message to the object along with an argument containing the URL the callback is for, and another containing a string which describes the current status. The status argument will consist of one of the following:

queued: on hold until a previous request to the same site is completed
contacted: the site has been contacted but no data has been sent or received yet
requested: the URL has been requested
loading *bytesTotal,bytesReceived*: the URL data is being received
uploading *bytesTotal,bytesReceived*: the file is being uploaded to the URL

Xserve



Density optimized rack mounted Mac OS X Server



UNIX-based Server Solutions from Apple

Nearly half a terabyte of storage per 1U

630 gigaflops of processing power

Hot-Swappable drives

Industry-standard 1U rack-optimized design

There has never been a better time to buy...

Small Dog Electronics carries factory refurbished models too offered at substantial savings!

(subject to availability)

Small Dog Electronics

www.smalldog.com

1673 Main Street, Route 100, Waitsfield, Vermont

Apple Specialist

To learn more: http://www.smalldog.com/xserve/

downloaded: the application has finished downloading the URL
uploaded: the application has finished uploading the file to the URL
error: an error occurred and the URL was not transferred
timeout: the application timed out when attempting to transfer the URL
(empty): the URL was not loaded, or has been unloaded

Since so much of the user interface that needs to be updated during a transfer is common to both uploading and downloading, I wrote one handler to handle status callbacks for both. Using a **switch** block to handle each type of status message appropriately, this handler was effectively the core of the application:

```
-
- UpdateStatus
-
- Callback from libURL providing status info
- and manages each stage of the transaction
- based on the current status
-
on UpdateStatus pUrl, pStatus
   set the itemdel to "/"
   put last item of pUrl into tFileName
   set the itemdel to comma
   put empty into tStatusDisplayString
   --
   switch item 1 of pStatus
      -- Handle progress:
   case "uploading"
      put "Uploading" into tStatusDisplayString
   case "loading"
      -- Update progress bar:
      put item 2 of pStatus into tBytesReceived
      put item 3 of pStatus into tTotalBytes
      set the endvalue of scrollbar "progress" \
        to tTotalBytes
      set the thumbpos of scrollbar "progress" \
        to tBytesReceived
      show scrollbar "progress"
      --
      if tStatusDisplayString is empty then
        put "Downloading" into tStatusDisplayString
      end if
      put cr& tFileName &cr& \
         Bytes2Size(tBytesReceived, "pad") & \
         " of "& Bytes2Size(tTotalBytes, "pad") \
         after tStatusDisplayString
      PutStatus tStatusDisplayString, "right"
      break
      --
      -- Errors:
   case "error"
   case "timeout"
      answer pStatus
      exit to top
      break
      --
      -- Completed successfully:
   case "uploaded"
      UpdateConnectionUI "connected"
   case "downloaded"
      hide scrollbar "progress"
      PutStatus "Done"
      enable btn "Download..."
      enable btn "Upload..."
      unload pUrl
      break
   end switch
end UpdateStatus
```

I added scripts to the **Upload** and **Download** buttons that simply call these handlers to initiate each respective transfer:

```
-
- UploadFile
-
- Called from the Upload button to select a local
- file and start uploading it to the server
-
on UploadFile
   answer file "Select a file to upload:"
   if it is empty then exit to top
   put it into tSource
   --
   set the itemdel to "/"
   put ServerDirectoryUrl()&last item of tSource into tDest
   --
   libUrlSetStatusCallback "UpdateStatus", long name of me
   libUrlFtpUploadFile tSource, tDest, "UpdateStatus"
   disable btn "Download..."
   disable btn "Upload..."
end UploadFile

-
- DownloadFile
-
- Called from the Download button to start downloading
- the selected file
-
on DownloadFile pFile
   ask file "Save this file to:" with pFile
   if it is empty then exit to top
   put it into tDest
   --
   put ServerDirectoryUrl()&pFile into tSource
   --
   libUrlSetStatusCallback "UpdateStatus", long name of me
   libUrlDownloadToFile tSource, tDest, "UpdateStatus"
   disable btn "Download..."
   disable btn "Upload..."
end DownloadFile
```

I had a little extra time so I added a useful flourish: I placed an animated GIF version of the flag over the image at the top of the window, and modified the UpdateConnectionUI handler to show the GIF when connected and hide it when disconnected. It was a small touch, but the animated element helped visually reinforce when the connection was "live".

After a little debugging to address a few typos, it seemed to be working well, so now it was time to build and test the standalone application.

3:30PM One of Revolution's greatest strengths is its broad support for deploying on multiple operating systems. With engines available for Mac Classic, OS X, all Win32 systems, and most flavors of UNIX and Linux it covers nearly every modern desktop system.

Standalones are built with Revolution's Distribution Builder (**Figure 4**), a utility accessed from the File menu, that lets you

MacTech® MAGAZINE
Get MacTech delivered to your door at a price FAR BELOW the newsstand price. And, it's RISK FREE!
Subscribe Today!
www.mactech.com

assign the application's file name and version info, assign icons, etc.. Once you set it up, you can save the settings for future use.

Figure 4. *Distribution Builder*

You just click the **Build Distribution** button and it does the rest, embedding the appropriate engine into a copy of your stack file to create the standalone. For OS X it constructs the bundle and writes the info.plist file for you as well.

Figure 5 shows the completed application in action.

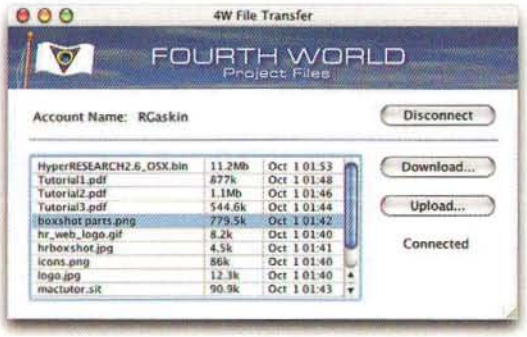


Figure 5. *Final application*

4:00 PM I compressed the application with SuffIt and emailed it to the client. He ran a few tests and was delighted with its simplicity. I left work early.

Conclusion

4GLs may not be the best choice for every job, especially computationally intensive tasks like writing device drivers or rendering 3D. But, when time-to-market is a concern a good 4GL like Revolution can be hard to beat. How many other systems let you build a custom FTP tool in under 200 lines of code?

If you want to check out the source for yourself you can download it from:

http://www.fourthworld.com/mactech/

Since the application requires your server info you'll need Revolution to set up the Admin window and build the application. You can download the free trial version of Revolution from Runtime Revolution Ltd.'s site:

http://www.runrev.com/

BMS

THE LAW OFFICE OF
BRADLEY M. SNIDERMAN

Need help safeguarding your software?

If you're developing software, you need your valuable work protected with trademark and copyright registration, as well as Non Disclosure Agreements.

Then, when you are ready to sell it, you can protect yourself further with a licensing agreement.

I am an attorney practicing in Intellectual Property, Business Formations, Corporate, Commercial and Contract law.

Please give me a call or an e-mail. Reasonable fees.

23679 Calabasas Rd. #558 • Calabasas, CA 91302
PHONE 818-222-0365 FAX 818-591-1038 EMAIL brad@sniderman.com

By Tim Monroe

Krakatoa, East of Java

Developing QuickTime Applications with Java

Introduction

Java is an object-oriented programming language and set of associated class libraries developed by Sun Microsystems in the early- to mid-1990's. It was designed and written largely by James Gosling, who sought to provide a simpler, more secure version of C++. The Java designers began with a syntax based on the C programming language (to promote familiarity with the new language among existing developers) but eliminated elements that promoted unstructured code (like the goto statement) or increased the likelihood of programming error or system misuse (like pointer arithmetic). The result was a clean, simple language that allowed developers an easy migration path from the world of procedural programming into the world of object-oriented programming. Java virtual machines — the runtime engines for compiled Java code — have been developed for a wide array of operating systems and devices.

QuickTime for Java is a set of Java classes and methods that implement large parts of the QuickTime multimedia architecture. Introduced in 1998 at the JavaOne conference, it can be used to develop standalone applications and applets (that is, code that runs within a larger host application, such as a web browser) that harness QuickTime's multimedia capabilities. Because they require QuickTime, QuickTime for Java applications and applets can run only on Macintosh and Windows computers.

In this article and the next two articles, I want to take a look at using QuickTime for Java to develop QuickTime applications. As in the past few *QuickTime Toolkit* articles, I want to see how to build a multi-window movie playback and editing application. Let's call this application "JaVeez". I also want to investigate ways to extend our application to handle potentially more complicated tasks. For the moment we'll focus solely on building an application that runs on Mac OS X. After we've done that, we'll take a look at the kind of changes we need to make in order for JaVeez to run on Windows operating systems as well.

Throughout these articles, we'll be using the latest released versions of Java and QuickTime for Java. At the time of this writing, the current version of the Java runtime engine on Mac OS X is Java 2 Standard Edition (J2SE) version 1.4.1, which was released in early 2003. This version incorporates a number of changes that allow applications to conform more closely to the standard Mac OS X Aqua look-and-feel. In particular, it allows applications to receive and respond to Apple events, which is essential (for instance) in allowing applications to open files dropped onto the application icon. We'll also rely on the version of QuickTime for Java included with QuickTime 6.4, which is the first release of this product that supports J2SE 1.4.1 on Mac OS X. (The version number of this new QuickTime for Java is 6.1.) The differences between this version of QuickTime for Java and earlier versions are substantial, but here I'm more interested in seeing how things are done using the current versions of these tools than in enumerating the precise changes from earlier versions.

We'll begin this article by creating a new project based on the Java AWT application template project provided by the Xcode development environment. We'll modify that project as necessary to support opening QuickTime movie files and displaying their movies in windows on the screen. Then we'll see how to create the application's menus and menu bar, and how to handle a few of the menu items in those menus.

In the next article, we'll continue working on JaVeez. We'll add the ability to edit movies and to save edited movies into their movie files. We'll also see how to support the standard document-related behaviors (such as prompting a user to save or discard changes to an edited file when the movie window is closed).

The Project

So let's get started. Launch Xcode and select "New Project..." in the File menu. In the list of available projects, scroll down to find the Java projects and then select "Java AWT Application", as in **Figure 1**. Name the new project "JaVeez" and save it in any location you like.

Tim Monroe is a member of the QuickTime engineering team. You can contact him at monroe@mactech.com. The views expressed here are not necessarily shared by his employer.

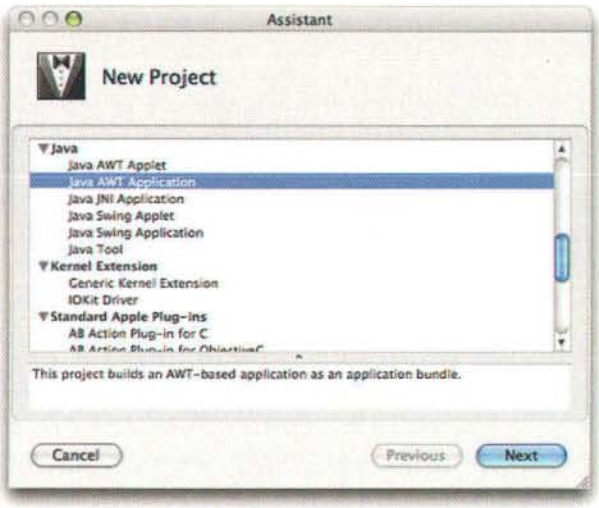


Figure 1: *The list of available Java projects*

AWT (which is short for "Abstract Window Toolkit") is a set of Java classes for creating and managing an application's user interface. It allows us to create windows, dialog boxes, menus, scrollbars, text labels, and so forth, using code that is platform-independent. AWT also provides a framework for handling events on items in the application's user interface.

The main official alternative to AWT is a set of classes called *Swing*. Swing is built on top of AWT and in many cases provides greater functionality than pure AWT. For instance, it's not possible, using AWT, to set the window modification state (so that the close button of a window whose document has been edited is drawn with a dot inside, as in **Figure 2**). It's fairly easy to do this in Swing, however. Similarly, Swing provides classes to display help tags (also called tool tips) on objects in the user interface, while AWT does not.

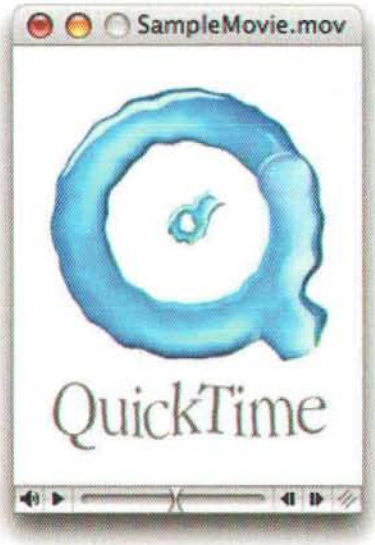


Figure 2: *A modified movie window*

For this reason and others, Apple generally recommends that Mac OS X Java applications be built using Swing window components instead of AWT window components. (In Java parlance, a *component* is any object that can be drawn on the screen and become the target of user actions.) However, QuickTime for Java does not easily support embedding a movie inside of a Swing component, if we want to attach a movie controller to that movie. So we'll use AWT to handle our application's movie windows and menus. In the next article, though, we'll see how to work with a few Swing components.

Modifying the Project

Once we've given our new project a name and a location, the new project window opens (**Figure 3**).

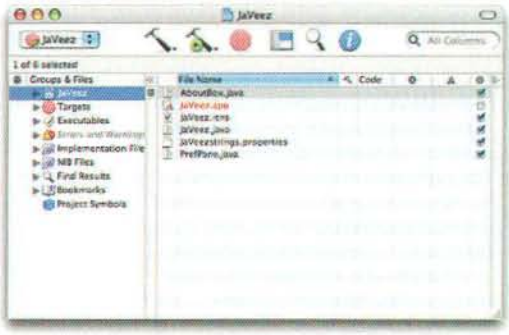


Figure 3: *The new project window*

HelpLogic
The Help Authoring Solution for Mac Developers
Project: Stimulus Help
Add File
Edit Page
Add Topic
Edit Topic
Preview
Publish
Project Files
fileformats.html
images
introduction.html
license.html
Help Table of Contents
Introduction
System Requirements
How to Use Stimulus
Supported Formats
Audio
Bass and Treble
Audio Equalizer
Workshop
Notes
To Do Items
Testing
Bug Reports
Easily create help systems for your software applications & web sites from a single source.
Save time with the integrated Workshop, TOC Builder, HTML Editor & Page Templates to quickly generate Apple Help, Web-based Help, UniHelp, PDF & more.
SNEAK PREVIEW
www.ebutterfly.com
electric butterfly

As you can see, there are three files with the filename extension ".java"; these are the source code files for this project. Let's go ahead and remove the files PrefPane.java and AboutBox.java, because our application will not support setting any preferences and because we'll develop a better way to handle our application's About box (in the next article).

Next, we need to add a file to the project. Select "Add Frameworks..." in the Project menu and navigate to the System/Library/Java/Extensions folder. Then select the file QTJava.zip. This file contains the QuickTime for Java packages that we'll need to use in our application. To make those packages available in our application, we need to import them. Add these lines near the top of the file JaVeez.java, after any existing import statements.

```
import quicktime.*;
import quicktime.io.*;
import quicktime.qd.*;
import quicktime.std.*;
import quicktime.std.clocks.*;
import quicktime.std.movies.*;
import quicktime.app.view.*;
```

The first non-import statement in this file is the beginning of the declaration of the JaVeez class:

```
public class JaVeez extends Frame {
```

This indicates that JaVeez is a subclass of (or *extends*) the AWT class Frame, which is the class for top-level windows with title bars and borders. Our movie windows will be instances of this class.

Immediately following the class declaration, you'll find declarations of class variables and instance variables. Here are the class variables we want JaVeez to support:

```
private static int nextHorizPos = 50;
private static int nextVertPos = 50;
private static Application fApplication = null;
private static ResourceBundle resBundle = null;
private static boolean launchedFromDrop = false;
```

There will be only one copy of each class variable, no matter how many instances of the JaVeez class our application creates (that is, no matter how many windows it opens). On the other hand, each instance of the class will get its own set of instance variables. Here are the ones we'll need to use:

```
private Movie m = null;
private MovieController mc = null;
private OpenMovieFile omf = null;
private QTComponent qtc = null;
private FileDialog fd = null;
private String baseName = null;
```

We'll learn what each of these variables does as we go along.

Starting Application Execution

A Java application begins execution in its main function, which is declared like this:

```
public static void main (String args[]) { }
```

In JaVeez, we'll ignore the args parameter, which contains the command-line arguments specified by the user if the application is launched on the command line. The first thing we need to do is initialize QuickTime. We'll call the open method of the QTSession class, but only if QuickTime has not already been initialized:

```
if (QTSession.isInitialized() == false)
  QTSession.open();
```

(This check is probably overkill for an application, but not for applets.) QTSession provides methods to initialize QuickTime and to provide information about the current operating environment. You must call its open method before using any other QuickTime for Java class.

If the QuickTime initialization completes successfully, we want to create a new empty movie window. We do this by calling the JaVeez constructor and passing it an empty string. Then we initialize the new frame by calling the method createNewMovieFromFile and display the frame to the user. If the user launched the application by dropping one or more movie files onto its icon, then we'll just hide that empty movie window. **Listing 1** shows the main method of JaVeez. (We saw just above that launchedFromDrop is a class variable that is initialized to false; we'll see the conditions under which it's set to true in the next article.)

Listing 1: Opening the application

main

```
public static void main (String args[]) {
  try {
    // initialize QuickTime, but not if it's already been initialized
    if (QTSession.isInitialized() == false)
      QTSession.open();

    // make an empty movie window
    JaVeez jvz = new JaVeez("");
    jvz.createNewMovieFromFile(null, false);
    jvz.toFront();

    // hide the movie if the application was opened by a dropped movie file
    if (launchedFromDrop)
      jvz.setVisible(false);

  } catch (QTException err) {
    // close down QuickTime session if an exception was generated
    err.printStackTrace();
    QTSession.close();
  }
}
```

If an exception is thrown, we'll call the close method of the QTSession class and exit the application.

Creating a New Window

The constructor method for the JaVeez class is quite simple, as you can see in **Listing 2**.

Listing 2: Constructing a new frame object

JaVeez

```
public JaVeez (String title) {
  super(title);

  // get the resource bundle
  if (resBundle == null)
    resBundle = ResourceBundle.getBundle("JaVeezstrings",
                                         Locale.getDefault());

  createActions();
  addMenus();
  createApplicationObject();

  // turn off resizing
  setResizable(false);
}
```

First, the constructor loads a *resource bundle* named "JaVeezstrings"; in JaVeez, this bundle contains a list of strings that specify menu titles, menu item titles, and the like. By loading strings from a resource bundle, we avoid having to hard-code them in our source code and thus facilitate localizing the application. For instance, when we build our menus, we retrieve the label for the New menu item in the File menu like this:

```
resBundle.getString("newItem")
```

You can look into the file JaVeezstrings to see what strings are defined therein.

After loading the resource bundle, we call three methods defined by JaVeez to set up the application's menus and menu-handling logic. Then we set the window so that it cannot be resized by the user. For simplicity, a movie window created by our application JaVeez will be set to a size that exactly contains the movie and the movie controller bar (if it's visible).

Initializing a New Movie

Most of the work required to display a QuickTime movie in an AWT frame is handled by our createNewMovieFromFile method, which is usually called immediately after the JaVeez constructor (as in **Listing 1** above). We pass createNewMovieFromFile the full pathname of the file to open, or an empty string if we want the window to contain a new, empty movie. To elicit a pathname from the user, we can use the standardGetFilePreview method of the QTFile class, as follows:

```
QTFile qtf = QTFile.standardGetFilePreview
                        (QTFile.kStandardQTFileTypes);
JaVeez jvz = new JaVeez(qtf.getPath());
jvz.createNewMovieFromFile(qtf.getPath(), false);
```

The first line of code displays the standard file-opening dialog box, shown in **Figure 4**:

Figure 4: *The file-opening dialog box*

Passing kStandardQTFileTypes to standardGetFilePreview indicates that we want the user to be able to select any type of file that QuickTime can open.

The createNewMovieFromFile method opens the specified file for reading and writing by creating a QTFile object and then passing that object to the asWrite class method of the OpenMovieFile class:

```
QTFile qtf = new QTFile(theFullPath);
omf = OpenMovieFile.asWrite(qtf);
```

If these methods succeed, createNewMovieFromFile calls the Movie constructor to create a movie object from that movie file and the MovieController constructor to create a movie controller object associated with that movie object. The Movie and MovieController classes are wrappers for QuickTime movies and movie controllers. Once we've opened a movie in a new window, most of our subsequent operations on the movie will be accomplished using methods supplied by the MovieController class.

But we still need to embed the QuickTime movie into the AWT frame. QuickTime for Java defines the class QTComponent, which represents displayable QuickTime objects. We create an instance of that class by calling the makeQTComponent factory method, and we then add that instance to the AWT frame by executing the frame's add method:

```
qtc = QTFactory.makeQTComponent(mc);
add(qtc.asComponent());
```

Our instance variable qtc is of type QTComponent, but add requires a parameter of type Component. As you can see, we call the asComponent method to get an AWT representation of the QTComponent. (If you are using Swing, you should create a QTJComponent; however, as mentioned earlier, there is no QTJComponent constructor that accepts a movie controller. That's the main reason we are using AWT components for our basic movie windows.)

The createNewMovieFromFile method then enables editing and keyboard control of the movie, using methods in the MovieController class. It finishes up by moving the movie window to the next staggered position on the screen. **Listing 3** shows our complete definition of createNewMovieFromFile.

Listing 3: Opening a movie file

createNewMovieFromFile

```
public void createNewMovieFromFile
        (String theFullPath, boolean useExistingWindow) {

  // set the window title
  baseName = basename(theFullPath);
  setTitle(baseName);

  try {
    if (theFullPath != null) {
      QTFile qtf = new QTFile(theFullPath);

      omf = OpenMovieFile.asWrite(qtf);
      m = Movie.fromFile(omf);
    } else {
      m = new Movie();
    }

    // create the movie controller
    mc = new MovieController(m);

    // create and add a QTComponent if we haven't done so yet;
    // otherwise set the movie controller
    if (qtc == null) {
      qtc = QTFactory.makeQTComponent(mc);
      add(qtc.asComponent());
    } else {
      qtc.setMovieController(mc);
    }

    // enable editing (unless movie is interactive) and key handling
    if ((mc.getControllerInfo() &
            StdQTConstants.mcInfoMovieIsInteractive) == 0)
      mc.enableEditing(true);

    mc.setKeysEnabled(true);

    // set the initial state of the menus
    adjustMenuItems();

    if (!useExistingWindow) {
      // set initial location of the movie window
      setLocation(nextHorizPos, nextVertPos);
      nextHorizPos += 20;
      nextVertPos += 20;
    }

    // set the size of the enclosing frame to the size of the incoming movie
    pack();
    setVisible(true);

  } catch (QTException err) {
    err.printStackTrace();
  }
}
```

You might be wondering why be didn't just add all this code to the constructor of the JaVeez class. The main reason for breaking it out into a separate method is that that allows us to reinitialize an existing movie window from a different movie file. We'll need to do this when we handle the "Save As..." menu item in the next article.

Setting the Title of a Window

Listing 3 calls the basename method to get the base name of a movie file (that is, the portion of the full pathname that follows the rightmost path separator). It uses that name to set the window title. The basename method is defined in **Listing 4**.

Listing 4: Getting the base name of a pathname

basename

```
public String basename (String pathName) {
  if ((pathName == null) || (pathName.length() == 0))
    return(resBundle.getString("newMovieName"));

  // if we are passed a full pathname, trim it to the last segment
  File file = new File(pathName);

  return(file.getName());
}
```

We return the default name for an empty movie file (which we read from the application's resource bundle) if the string passed into the method is null or an empty string. Otherwise, we call the getName method of a File object to get the name of the specified file. As you saw in **Listing 3**, we store the movie's returned base name in an instance variable so that we can use it in the method that displays the standard "Save Changes" dialog box, as we'll see in the next article.

Setting the Size of a Window

The pack method called in the createNewMovieFromFile method sets the size of the content area of the frame object to

the size of the movie that was just opened, including the rectangle occupied by the movie controller bar (if visible). Occasionally, we'll need to adjust the size of the movie window, even though we don't allow the user to resize it manually. For instance, when the user cuts a segment from a movie, the size of the movie may change. In that case, we'll call our own method sizeWindowToMovie (**Listing 5**) to resize the movie window.

Listing 5: Setting the size of a movie window

sizeWindowToMovie

```
public void sizeWindowToMovie () {
  try {
    QDRect rect = m.getBox();

    if (mc.getVisible())
      rect = mc.getBounds();

    // make sure that the movie has a non-zero width;
    // a zero height is okay (for example, with a music movie with no controller bar)
    if (rect.getWidth() == 0) {
      rect.setWidth(this.getSize().width);
    }

    // resize the frame to the calculated size, plus window borders
    setSize(rect.getWidth() +
                (getInsets().left + getInsets().right),
            rect.getHeight() +
                (getInsets().top + getInsets().bottom));
  } catch (QTException err) {
    err.printStackTrace();
  }
}
```

As you can see, we just use the MovieController method getBounds to get the size of the movie and controller bar; then we add in the heights and widths of the window borders.

Menus

Creating menus and handling user selection of menu items in Java applications is reasonably straightforward. Both AWT and Swing provide classes from which we can instantiate menu bars, menus, and menu items. The only "gotcha", at least for those of us who cut our programming eyeteeth on the Macintosh, is that Java menu bars are attached to individual frames — that is, to individual windows. That means that if no movie window is open, then JaVeez' menu bar won't contain any menus other than the Application menu, which is provided automatically by the operating system. **Figure 5** shows this minimal menu bar.

Figure 5: *The JaVeez menu bar when no movie windows are open*

This is not an ideal situation. For one thing, it means that if the user closes all the open movie windows, the File menu disappears and there is no way to open additional movies via the menu bar. (A clever user could of course drag a movie file onto the application's icon in the Finder or in the dock.) Still, it's not a situation worth worrying too much about, since there is an easy workaround: when the application is launched, just open an empty window and move it to an offscreen location where it will not be visible. (Implementing this simple workaround is left as an exercise for the reader.)

As I said, both AWT and Swing will allow us to create menu bars, menus, and menu items. Since we're already using an AWT frame for the movie window, let's continue down that path and use the AWT menu classes. Swing does not offer any additional menu-related capabilities that we need to use in JaVeez.

Creating Actions

When the user selects an item in a menu, the Java runtime engine sends an *action event* (which is an object of type ActionEvent) to the menu item. The menu item in turn passes the event to any registered listeners. These listeners are *actions* (of type Action). So the first thing we need to do is create an action for each menu item in our application.

To create an action object, we define a concrete subclass of the AbstractAction class. This subclass must implement the actionPerformed method. **Listing 6** gives our definition of the NewActionClass class, which will be instantiated to handle the New menu item.

Listing 6: Handling the New menu item

NewActionClass

```
public class NewActionClass extends AbstractAction {
  public NewActionClass (String text, KeyStroke shortcut) {
    super(text);
    putValue(ACCELERATOR_KEY, shortcut);
  }
  public void actionPerformed (ActionEvent e) {
    JaVeez jvz = new JaVeez("");
    jvz.createNewMovieFromFile(null, false);
    jvz.toFront();
  }
}
```

Similarly, **Listing 7** gives our definition of the OpenActionClass class, which will be instantiated to handle the Open... menu item.

Listing 7: Handling the Open menu item

OpenActionClass

```
public class OpenActionClass extends AbstractAction {
  public OpenActionClass (String text, KeyStroke shortcut) {
    super(text);
    putValue(ACCELERATOR_KEY, shortcut);
  }
  public void actionPerformed (ActionEvent e) {
    try {
      QTFile qtf = QTFile.standardGetFilePreview
                          (QTFile.kStandardQTFileTypes);

      JaVeez jvz = new JaVeez(qtf.getPath());
      jvz.createNewMovieFromFile(qtf.getPath(), false);
      jvz.toFront();
```

```
    } catch (QTException err) {
      if (err.errorCode() != Errors.userCanceledErr)
        err.printStackTrace();
    }
  }
}
```

Both of these class implementations call the method putValue to associate the action with a keystroke combination, which (as we'll see shortly) is passed to the class constructor. JaVeez declares AbstractAction subclasses for each of its dozen or so menu items. In the interest of saving space, I've omitted the remaining definitions.

Once we've defined a concrete subclass of AbstractAction for each menu item, we need to create actions for each such subclass. JaVeez declares instance variables for all of these actions:

```
protected Action newAction, openAction, closeAction,
      saveAction, saveAsAction;
protected Action undoAction, cutAction, copyAction,
      pasteAction, clearAction, selectAllAction,
      selectNoneAction;
protected Action toggleBarAction, toggleSpeakerAction;
```

We create actions by invoking the class constructors. **Listing 8** shows how we do this for three of these actions. Once again, the code for the remaining cases has been omitted in the interest of brevity.

Listing 8: Creating actions

createActions

```
public void createActions () {
  int shortcutKeyMask =
      Toolkit.getDefaultToolkit().getMenuShortcutKeyMask();

  // create actions that can be used by menus, buttons, toolbars, etc.
  newAction = new NewActionClass(
              resBundle.getString("newItem"),
                KeyStroke.getKeyStroke(KeyEvent.VK_N,
                                       shortcutKeyMask));
  openAction = new OpenActionClass(
              resBundle.getString("openItem"),
                KeyStroke.getKeyStroke(KeyEvent.VK_O,
                                       shortcutKeyMask));

  // lots of lines omitted here...

  toggleBarAction = new ToggleControllerActionClass(
              resBundle.getString("hideControllerItem"),
                KeyStroke.getKeyStroke(KeyEvent.VK_1,
                                       shortcutKeyMask));
```

Creating Menus and Menu Items

Now that we've created the actions that will handle selections of menu items, we can proceed to create the menu items and insert them into menus. First, let's create the main menu bar, like this:

```
protected MenuBar mainMenuBar = new MenuBar();
```

A menu bar contains menus, which are objects of type Menu. JaVeez has three application-specific menus: the File menu, the Edit menu, and the Movie menu. We'll use these instance variables to refer to them:

```
protected Menu fileMenu;
protected Menu editMenu;
protected Menu movieMenu;
```

Listing 9 shows our definition of the addMenu method, which creates these menus and their items and then adds them to the menu bar. It also sets mainMenuBar as the menu bar for the frame under construction.

Listing 9: Configuring the menu bar

addMenus

```
public void addMenus () {
  editMenu = new Menu(resBundle.getString("editMenu"));
  fileMenu = new Menu(resBundle.getString("fileMenu"));
  movieMenu = new Menu(resBundle.getString("movieMenu"));

  addFileMenuItems();
  addEditMenuItems();
  addMovieMenuItems();

  setMenuBar(mainMenuBar);
}
```

All that remains is for us to write the addFileMenuItems, addEditMenuItems, and addMovieMenuItems methods. These methods create the individual menu items, set their keyboard shortcuts, add them to the appropriate menu, and then attach the action listeners created earlier. **Listing 10** shows the complete definition of the addFileMenuItems method, which uses these instance variables:

```
protected MenuItem miNew;
protected MenuItem miOpen;
protected MenuItem miClose;
protected MenuItem miSave;
protected MenuItem miSaveAs;
```

Listing 10: Adding menu items to the File menu

addFileMenuItems

```
public void addFileMenuItems () {
  miNew = new MenuItem(resBundle.getString("newItem"));
  miNew.setShortcut(new MenuShortcut(KeyEvent.VK_N,
                                     false));
  fileMenu.add(miNew).setEnabled(true);
  miNew.addActionListener(newAction);

  miOpen = new MenuItem(resBundle.getString("openItem"));
  miOpen.setShortcut(new MenuShortcut(KeyEvent.VK_O,
                                      false));
  fileMenu.add(miOpen).setEnabled(true);
  miOpen.addActionListener(openAction);

  miClose = new MenuItem(resBundle.getString("closeItem"));
  miClose.setShortcut(new MenuShortcut(KeyEvent.VK_W,
                                       false));
  fileMenu.add(miClose).setEnabled(true);
  miClose.addActionListener(closeAction);

  fileMenu.addSeparator();

  miSave = new MenuItem(resBundle.getString("saveItem"));
  miSave.setShortcut(new MenuShortcut(KeyEvent.VK_S,
                                      false));
  fileMenu.add(miSave).setEnabled(false);
  miSave.addActionListener(saveAction);
```

MacTech®

MAGAZINE

The source for developers and the technical market since 1984.

CRASH PROOF

- No installer needed
- No download times
- Readable in your favorite reading locations
- No drain on your system's resources
- The best reader user interface mechanism
- Available Monthly
- Delivered painlessly to your doorstep.

Authored by a number of industry experts working in the real world

Toll Free: 877-MACTECH

Get it today RISK FREE at http://www.mactech.com

```
    miSaveAs = new MenuItem
                (resBundle.getString("saveasItem"));
    miSaveAs.setShortcut(new MenuShortcut(KeyEvent.VK_S,
                                          true));
    fileMenu.add(miSaveAs).setEnabled(true);
    miSaveAs.addActionListener(saveAsAction);

    mainMenuBar.add(fileMenu);
}
```

Notice that we call the addSeparator method to insert a menu separator into the menu. **Figure 6** shows the resulting File menu.

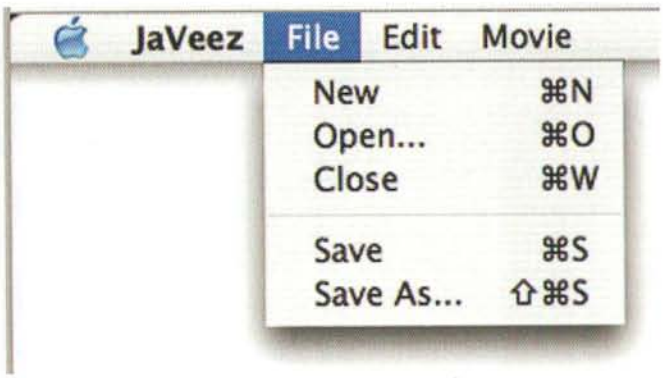


Figure 6: *The File menu of JaVeez*

Movie Playback

So, we've managed to open a movie file in a window, appropriately sized to exactly contain the movie at its natural size and the associated movie controller bar (if it's visible). **Figure 7** shows a movie window displayed by JaVeez. As you can see, there is no grow button in the movie controller bar and the zoom button in the title bar is disabled; both of these result from our decision to disallow manual movie window resizing.

MacTech MAGAZINE

Get MacTech delivered to your door at a price FAR BELOW the newsstand price. And, it's RISK FREE!

Subscribe Today!
www.mactech.com

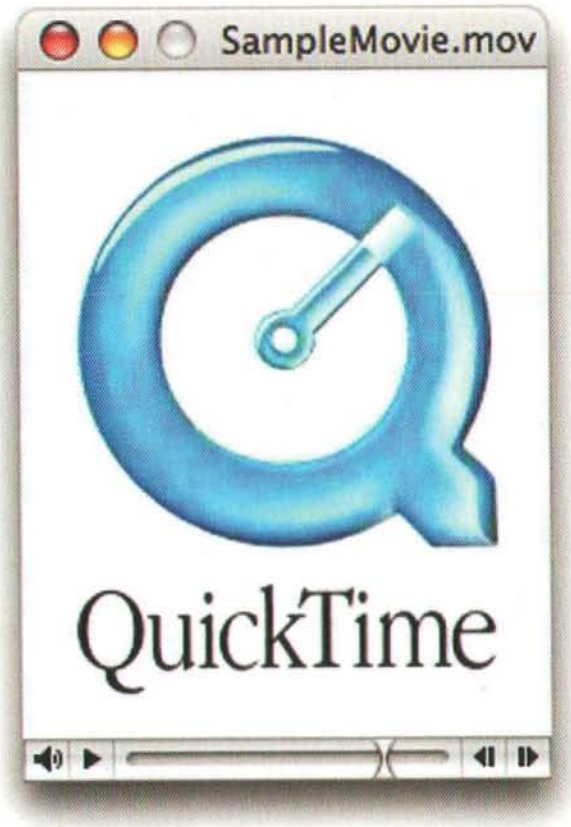


Figure 7: *A JaVeez movie window*

AWT handles all the low-level nitty-gritty of displaying and managing the open movie windows. It handles dragging windows around, as well as iconifying (that is, minimizing) and deiconifying them. And the MovieController object handles most events that occur within the window frame. It handles mouse clicks within the movie and, for QuickTime VR movies, zooming in and out using the Shift and Control keys.

Nonetheless, the movie controller is neglecting to handle some events that, in theory, it ought to be handling. It does not start or stop a linear movie when the spacebar is pressed, and it does not pan or tilt a QuickTime VR movie when the arrow keys are pressed. This is a bug in QuickTime for Java 6.1, which will be fixed in a future release. In the meantime, it's easy enough to work around this misbehavior. In this section, we'll see how to do that, and also how to handle the "Hide Controller Bar" menu item in the Movie menu.

Handling Keys

To get the movie controller to process key events, we can have the JaVeez class implement the key listener interface. To do this, we'll change the declaration of JaVeez slightly, so that it looks like this:

```
public class JaVeez extends Frame implements KeyListener {
```

Then we need to provide implementations of each of the methods defined in that interface. There are three such methods: keyPressed, keyReleased, and keyTyped. The keyPressed method is invoked when a key is pressed; the keyReleased method is invoked when a key is released; the keyTyped method is invoked when a key is pressed and then released. For our purposes, we want to implement the keyPressed method, shown in **Listing 11**. (The remaining two methods are empty.)

Listing 11: Handling key-pressed events

keyPressed

```
public void keyPressed (KeyEvent e) {
  try {
    mc.key(e.getKeyCode(), e.getModifiers());
  } catch (QTException err) {
    err.printStackTrace();
  }
}
```

We simply pass the key code and the key modifiers to the key method of the MovieController. Problem solved.

Handling the Movie Menu

It's also quite easy to hide or show the movie controller bar. When the user selects the "Hide Controller Bar" menu item, JaVeez executes the method defined in **Listing 12.**

Listing 12: Toggling the visibility state of the controller bar

actionPerformed

```
public void actionPerformed (ActionEvent e) {
  try {
    mc.setVisible(!mc.getVisible());
    sizeWindowToMovie();
    adjustMenuItems();
  } catch (QTException err) {
    err.printStackTrace();
  }
}
```

We'll take a look at the adjustMenuItems method in the next article. In part, it changes the menu item text to reflect the current state of the controller bar visibility.

Conclusion

In this article, we've seen how to develop a basic Java application that can open one or more QuickTime movie files and display their movies in windows on the screen. We'll continue developing JaVeez — by adding the ability to edit movies and then save those edited movies into their files — in the next article.

Acknowledgements

Thanks are due to Anant Sonone and Tom Maremaa for reviewing this article and providing some helpful comments. Special thanks are also due to Chris Adamson (of Subsequently and Furthermore, Inc.) and Daniel H. Steinberg (of Dim Sum Thinking, Inc.) for their assistance and support.

Expand your LAN

TrangoLINK-10™ Outdoor Wireless Ethernet Bridge

- 10 Mbps up to 40 Miles
- Easy to Install
- Easy to Manage
- Secure

Expand your existing LAN in hours! The TrangoLINK-10 point-to-point outdoor wireless Ethernet bridge offers secure, high-speed transmission in the license exempt 5.8 GHz and 5.3 GHz bands. Ideal for building-to-building connectivity, the TrangoLINK-10 can be integrated into existing networks for a low-cost, scalable LAN expansion solution.

trangobroadband WIRELESS

www.trangobroadband.com
Phone: 858.653.3900
Email: sales@trangobroadband.com

Fixed Wireless that Works!

By Dave Wooldridge

Turning Users into Customers

Your Software Should be an Effective Selling Tool

In the last two installments of this column, we explored web site marketing and online publicity, but now it's time to shine the spotlight on your software. Is your product designed to be its own best sales agent or are *you* doing all the work? This month, we'll take an in-depth look at the benefits of trialware and demoware, the pros and cons of various approaches, and focus on simple strategies that can help your software sell itself!

Embracing a Buyer's Market

Ten years ago, delivery mechanisms for software were primarily limited to online BBS, newsgroups, floppy disk, and CD-ROM. Then the Internet evolved into a world-wide phenomenon, providing both shareware and commercial developers with an unprecedented distribution platform for reaching new customers on a global scale.

The online world has allowed consumers to easily research and educate themselves on the spectrum of products available to solve their software needs. They compare feature sets and prices and are now more equipped to make intelligent purchase decisions. Gone are the days of walking into a local CompUSA and blindly selecting from the choices on the shelf. In this day and age, consumers expect to try software before they buy it. It doesn't matter if your software is priced at $5 or $500. If they cannot test drive your software, but have the opportunity to download and evaluate your competition, then it's probably safe to assume you've just lost a potential sale.

The Internet has proven to be such a vital marketing vehicle for selling software that even the major heavyweights like Adobe, Macromedia, Apple, and Microsoft, who used to only offer boxed retail versions of their commercial software, are now also offering downloadable trial versions. The "try before you buy" revolution has become such a ubiquitous part of the software business that the definition of shareware has blurred beyond the point of recognition. With the Internet flooded with countless software titles vying for consumer attention, who can tell the difference anymore between commercial products and shareware? Lately, it seems like most downloaded software can be unlocked and registered with a purchased serial number and an Internet connection.

So what does this mean to you as a developer? No matter what kind of software you create, if you're making it available to the public, then you need to formulate a way for consumers to try it before they buy. It is an expectation that is hard to avoid. And why would you? Even though it initially creates more development work for you, it also provides a valuable marketing opportunity as well.

While many developers realize that having downloadable trial versions of their software is a necessary sales component, the most frequently asked questions are usually where to start and what approach to adopt? Should the software be offered as a demo with some features disabled or should the software be fully functional with a limit placed on time or usage? Are nag dialog messages effective? Should an installer be used?

A Quality User Experience

Let's start by looking at the overall package. The primary purpose of trialware is not to simply inspire interest, but to make the sale. It should convince users that they don't merely want your software, but that they *need* your software. Posting a demo on your site, compressed in a Stuffit archive with nothing more than a simple "Read Me" file may not be enough. All too often there is a developer mentality that if consumers are interested in the product, they can just buy it to access all of the bells and whistles, so no need to spend a lot of time on the demo. This is short-sighted thinking because consumers want to see the bells

Dave Wooldridge is the founder of Electric Butterfly (www.ebutterfly.com), the web design and software company responsible for Stimulus, HelpLogic, UniHelp, and the popular developer site, RBGarage.com.

and whistles upfront – they want to be impressed. Your competitors' products have similar feature sets and may even offer them at lower prices. Why should consumers choose your product? Give them a reason.

We've mentioned previously the importance of first impressions. Don't assume that consumers have thoroughly read the product information on your web site prior to downloading your demo. Don't even assume that they've ever visited your site. Your trial version could have been downloaded from VersionTracker.com, MacUpdate.com, a MacAddict CD, or any number of other places. A consumer may not have any idea what your product does, only that it had a cool name that sparked their interest. The first few minutes of exploring your demo will determine whether or not they place it in the trash or purchase a license.

With this in mind, your objective is to build a quality user experience from the second the product is downloaded to their desktop to the moment they select Quit from your application's menu.

Disk Images

For software that does not require an installer, Apple recommends distributing it as a Mac OS X disk image. Ever have customers complain that they downloaded your software, but couldn't remember where they saved it on their hard drive? With Apple's Safari web browser, downloaded disk images are automatically mounted on the desktop, providing an instant, easy access display that is very user friendly. Apple's Disk Copy utility (included with Mac OS X) can create disk images. If the disk image is saved as Read & Write, you can then tweak the window appearance and background graphics from options provided in the Finder's View menu. For software downloads, you probably won't want people messing with your disk image layout, so you'll want to create your disk image as Read Only. As of this writing, you cannot create Read Only disk images with customized background graphics using Disk Copy, but you can with MindVision FileStorm (http://www.mindvision.com/filestorm/).

Figure 1 reflects the kind of custom disk image you can create with MindVision FileStorm. Using our fictional CodeQuiver product as an example, we created a background image that conveys three distinct messages:

Brand Identity. CodeQuiver's name and logo are placed across the top of the disk image window. This not only promotes the product branding, but also gives the disk image a professional, polished look.

Product Description. How many times have you downloaded software, but didn't have time to try it until weeks later, at which point you'd forgotten what it was? Including a brief description or list of key features will remind users what your product does.

Instructions. Don't make users guess what to do next. Frustration will negatively overshadow their user experience. Avoid confusion by including a brief sentence on how to proceed with installing your software (even if it's as easy as dragging the application to the hard drive).

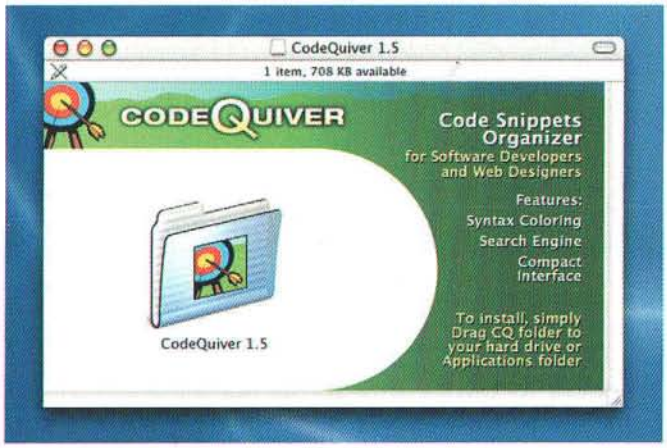


Figure 1. *An example disk image for our fictional CodeQuiver. Creative disk image backgrounds allow you to remind users what your product does and how to install it.*

Installers

If your application requires several support files to be installed in various locations, don't instruct users to do these steps manually. More often than not, consumers will not be bothered with a laborious installation and your download will be deleted. Convenience is the key factor here. You want users to be up and running with your application in a matter of minutes. Automate the installation task by using an installer package.

If you plan to use MindVision FileStorm to create your disk images, you may want to upgrade to their FileStorm Pro edition, which adds the ability to produce royalty-free installers. Like its disk image tools, FileStorm Pro can create installers with custom background images. This is a great way to not only promote brand identity, but it also gives you an opportunity to educate users on your product's benefits and key features while they wait for the installation to finish. Embedded in the background image of our fictional CodeQuiver installer (see **Figure 2**) we've included a "Did You Know?" section that reveals cool tips that new users may not be aware of. The more knowledgeable consumers are on how to use your product will greatly improve their first time experience running the application.

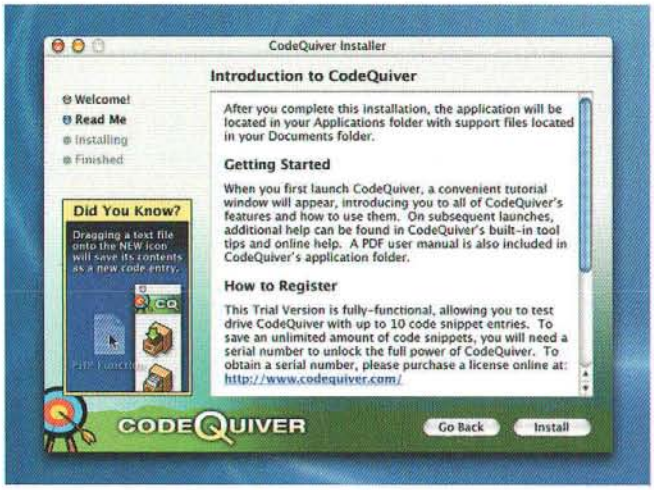


Figure 2. *Educate new users with cool product tips while they wait for the installation to finish.*

If your software requires more advanced installation options than what are currently offered in FileStorm Pro, you can check out other solutions such as MindVision Installer VISE (http://www.mindvision.com/), InstallFactory (http://www.installfactory.com/) or Stuffit InstallerMaker (http://www.stuffit.com/mac/installermaker/). To read Apple's Software Distribution FAQ or to download their own installer tools, visit the Apple Developer Connection online at http://developer.apple.com/.

Choosing a Path

After determining how you're going to package your download, the next step is creating the actual trial application. Here's where the road splits for many developers. Shareware fundamentalists believe that downloaded software should be free of restrictions, placing the responsibility on consumers to pay for the software they use. Wary of the increase in software piracy and file-sharing trends, a new breed of Internet-savvy developers believe that the current online environment demands a different approach. It's not that they think consumers are inherently dishonest, but why pay for something if it can be downloaded for free? With so much freeware available online, the Internet has created a scavenger mentality. In the minds of many consumers, not paying for downloaded software is never considered stealing, but saving money. To combat these misconceptions, more and more shareware titles are being released with some kind of restriction, whether it be based on time, usage or locked features, that force users to purchase a serial number to remove those restrictions.

If you depend on your software sales as a primary source of income, then it's in your best interest to find an effective strategy that will motivate users to purchase. As we explore the pros and cons of various concepts, keep in mind that there is no single "magic bullet" solution. You have to take a close look at which strategy will work best for your target audience and your application's feature set. For example, disabling the "Save" feature may work well for multi-featured applications like word processors or graphics editors, but if "Save" is the primary feature, such as in an audio converter utility, then the demo would be too crippled for users to properly evaluate. Your goal is to protect your software without sacrificing the user experience. Sales will not increase if users turn away frustrated.

Trialware

Trialware is usually defined as an application that grants access to all of its full-functioning features for a limited span of time or number of uses. Where traditional shareware titles simply ask users to purchase a license within 30 days, trialware actually enforces this policy by either locking down the entire program or disabling key features upon expiration.

Pros. The greatest strength of trialware is its ability to introduce users to every aspect of the application without compromise. None of the features are disabled, so for a limited time, users can properly evaluate the software as if it were the full registered version. To determine how many days or uses you should adopt for your trial version, try to calculate the average length of a user session, or if your application is project-based, figure out how long it might take a user to create a project. This should help narrow down an appropriate time frame since you want to give users enough time to adequately test drive your software without giving them so much time that they complete their task without needing to purchase. For example, if your software is a web site editor, a 30-day window may give users ample time to finish building their own web site, leaving little motivation for them to purchase if they consider the job to be done. In that scenario, reducing the time frame to only 10 days or 20 uses, etc. may provide enough evaluation time without "giving away the farm."

Cons. The one disadvantage to tracking a 30-day time limit or number of uses is that it leaves developers with the predicament of how to store this information so that it cannot be modified or hacked by users. A common approach is to save an invisible data file somewhere on the user's system (such as the Preferences folder), but laziness breeds creativity. To prevent a 30-day trial from expiring, many users simply roll back their computer clock to grant themselves more time. And if a developer avoids this situation by placing the restriction on the number of sessions, there are plenty of power users who have figured out how to locate these hidden files and change the data to extend the usage. An effective approach that bypasses these user hacks is to set a time limit on each user session. So instead of limiting the number of uses or the number of days, you simply limit the session length. When a user launches your application, an "unregistered" dialog message indicates

that the user has 30 minutes to test drive the application. Developers can program a built-in timer to monitor the session time. The key benefit to this approach is that the internal timer is part of the application, so it bypasses common user hacks since no invisible data file is needed. Of course, users can keep relaunching your software after each time it quits, but after sitting through the initial "unregistered" nag dialog with each successive launch, the continued inconvenience may provide the necessary motivation to purchase a license.

Figure 3. *TOP: A typical demo dialog message that appears when a user tries to access a disabled feature. BOTTOM: Provide more information with a call to action, encouraging the user to click the "Purchase" button.*

Demoware

Since many developers are wary of releasing fully-functional trialware, their sales motivators come in the form of disabled features. Demoware is typically fully-functioning except for one or two key features that are specifically disabled. Users are then prompted to purchase a serial number in order to activate those features. For games, usually the demo includes the first 1 to 3 levels with no restrictions on game play. If the user wants access to more levels, then the game must be registered with a serial number in order to unlock the additional levels. Some graphics and printing utilities place "unregistered demo" watermarks on printed and/or on-screen documents – this cripples the features while still leaving them enabled for proper evaluation.

If you display a dialog message when users try to select disabled features, don't waste the opportunity with a passive phrase like "this feature is disabled in the demo" (see **Figure 3, Top Example**). Extend the message to be more informative, stating exactly what is disabled and what a user would gain from purchasing a license (see **Figure 3, Bottom Example**). Instead

Long Distance

3.9¢ Per Minute!

Straight 6 second billing increments

Excellent rates on intrastate, intralata/toll calls and international calling with no term contract.

Toll Free (800/888/877/866) service, same low per minute rate for incoming calls.

10 cents per minute calling card.

Detailed billing directly from Capsule Communications, a Covista Company.

Quality electronic and telephone customr support.
No monthly billing fee if you sign up for AUTOPAY billing option or if your bill is over $20.00 each month.

(NOTE: $1.00 billing fee is charged when your bill is under $20.00 for all non-Autopay customers.)

www.lowcostdialing.com

of the usual "Okay" button that dismisses the dialog, include a call to action in your message with "Purchase" as the default button and "Not Now" as the cancel button. This forces the user to make a choice. It also adds another level of convenience since your "Purchase" button could link directly to either your online web store (such as Kagi, eSellerate, RegSoft, DigiBuy or your own custom e-commerce site) or your application's built-in order form (like eSellerate's Integrated eSeller). Avoid using the error or warning icons with these dialog messages since the user did nothing wrong.

Pros. There are two primary advantages that demoware holds over trialware. One is that demoware is much more difficult to hack. And two, demoware never expires. The problem with a 30-day trial is that if a user launches the application once and then doesn't have time to revisit the software until weeks later, the trial may expire before the user has a chance to successfully evaluate it. We're all busy and often find ourselves downloading trialware that sits on our desktop for weeks until we have the time to really give it a good spin. Since demoware's chief deterrent is in crippling key features, users never have to feel pressured for time since demos never expire.

Cons. A major drawback to disabling a feature is that it prevents users from previewing that feature. Choosing the feature that is crippled is often a very difficult task for developers. What if the feature you choose to disable is the very feature that some users want to test drive? If they care most about that one feature, but are unable to try it, you run the risk of losing those sales. You could allow users to select which features are disabled at runtime, but then that essentially exposes your program as fully-functional over the course of several back-to-back sessions, defeating the whole purpose of demoware.

Nagware

There are old school shareware veterans who still believe in offering unlimited, fully-functional software, relying on a consumer's sense of ethics to register the program if they decide to keep it. It's a traditional honor system at its finest with nothing but the occasional "nag" dialog message to remind consumers to do the right thing. "You've been using Program X for 12,593 days. Click here to register now." Does it work? Is it enough to motivate people to pay? If you're looking to make a full-time living selling software, then nagware is probably not the right vehicle for your products. In the current business climate, most developers find that nagware usually generates only enough sales to buy dinner and a movie on the weekends – affectionately referred to as "fun money." Of course, there are exceptions to this rule. Some products, such as the popular GraphicConverter, have done quite well with nothing more than "nag" dialog messages, but these rare exceptions are typically stellar, award-winning software that would likely excel in any marketing scenario.

Obviously, nagware is not going to net you a lot of sales if your product carries an expensive price tag, but don't assume lower priced software (under $20) is going to fare much better. While price does affect consumer purchase decisions, if there is no deterrent other than the occasional "nag" dialog message, then the defining issue becomes laziness, not price. Why register if they don't have to? If consumers are already using your software for "free," then you're going to have to give them a good reason why they should pay for it.

Successful nagware products (and even trialware) often "dangle carrots" (sales incentives) to motivate users to buy a license. Promise them extra features, free updates, unlimited support... but whatever you do, never treat your users like criminals. It's okay to lay on a little guilt about supporting shareware and future development, but your software's dialog messages and documentation should never make unregistered users feel like thieves.

Lending a Helping Hand

As developers, we know our own software creations so well that we often forget that new users will not be able to navigate through the interfaces with quite the same ease. While including documentation with the download (such as a PDF manual) is much appreciated by consumers, reading it will not be the first thing they do. How often do you read the electronic manual before double-clicking on the application icon? Never? Well, don't expect consumers to act any different. Anxious to try out your software, even well-placed "Read Me" files are generally overlooked. And adding multiple exclamation points with all capital letters may not improve the popularity of your ignored "READ ME!!!" file.

Even if your software includes built-in help (such as Apple Help), consumers typically regard that format as a last resort to solving problems. But you do not want them to struggle with your software until frustration finally leads them to the built-in help. At that point, they are so flustered and impatient that your help pages can't possibly calm their rattled nerves. People do not buy products that add confusion to their already hectic lives.

Good software should be focused not only on offering superior features, but also on providing a superior user experience. An integrated tutorial can be a great time-efficient solution, visually showing users how to operate the software without forcing them to read a lengthy manual. A built-in tutorial can be something as simple as a window with arrow buttons that take the user from screen to screen, illustrating step-by-step how to use the application interface to complete common tasks. Think of it as a digital quick start guide. Your software can automatically display the tutorial upon first launch of the

application, and then make it accessible at any time via the Help Menu (see **Figure 4**).

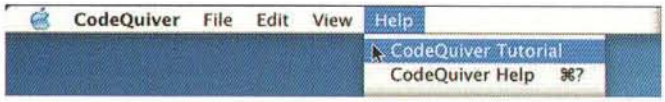


Figure 4. *Don't assume that new users know how to use your software. Beyond traditional online help, include an on-screen tutorial to help them navigate through the interface and key features.*

Another helpful method of quickly introducing new users to key features, keyboard shortcuts, and other benefits is through the use of a "tips" window (see **Figure 5**). As you've seen employed in many popular software products, a tips window is usually displayed as a small floating palette window that utilizes very little screen space. Curious users can cycle through the various tips by clicking the "Next Tip" button. The Preferences window can then provide an option to either enable or disable the tips window from appearing upon application launch.

Figure 5. *Including an optional "tips" window with little known facts, hidden features and keyboard shortcuts can greatly enhance a new user's experience.*

Yet another helpful resource: examples! If your software creates project-based files or complex documents, then include several sample files to showcase possible uses for specific features. This is especially important if either the "New" or "Save" feature is disabled since users can still open finished files for review. These examples can also demonstrate ways to use the application that consumers may not have originally thought of, increasing the perceived value of your software.

Be Creative

The two fundamental secrets to turning new users into customers are (1) make users happy, and (2) give them a reason to buy. Customized disk images, hassle-free installers, tutorials, tips, and example projects are just a handful of concepts for adding a professional polish to your trialware or demoware, but every application is different. Harness your creativity to develop an effective download solution that protects your product's salability while providing an easy-to-use, intuitive evaluation environment. You may have produced a great software product, but if the first impression of your downloaded trial version does not motivate users to pull out their credit cards, you might never get a second chance to win their love... or money.

MacTech®
MAGAZINE

Get MacTech delivered to your door at a price FAR BELOW the newsstand price. And, it's RISK FREE!

Subscribe Today!
www.mactech.com

stockicons.com
brought to you by the Iconfactory

Icons to go!

Now developers can purchase complete collections of professionally designed icons at an affordable price from the premier source for custom designed icons in the industry. See some of the work we have done for Apple, Microsoft, Aladdin, Intuit, Palm and many others at www.iconfactorydesign.com.

$349.00

Each stock icon collection contains 80 individual icons in three pixel sizes - 32x32, 24x24 and 16x16. Collections are unique in style and are provided in an array of formats including transparent TIFFs, GIF, PNG, .icns, and Windows format .ico's.

For more information visit www.stockicons.com

By David Breffitt

Alsoft's DiskWarrior 3

New version of an old standard

Introduction

You are bound to have problems sooner or later. Apple ships tools with the OS, but those in the know supplement those tools with third party solutions. Our favorite is DiskWarrior. I cannot tell you the amount of times that this product saved us, or solved the weirdest of seemingly unrelated problems.

Mac users need a disk repair tool on which they can consistently rely. Not only must it never lose the data that they have, but it must also be powerful enough to be able to get all of their data back. Plus, a tool that monitors for drives for hardware malfunctions would have enormous benefits for Mac users.

DiskWarrior is the only utility that takes a different approach to repairing directory damage. It doesn't try to repair the old directory. It simply builds a new and error free directory each time it is run.

How does the directory damage occur in the first place? The continued accuracy of information in the directory depends on the ability of the Mac OS to perform ALL of its update and maintenance operations without any expected interruptions. Also, to help speed up many Mac OS computer operations, important pieces of information are temporarily kept in memory instead of being immediately saved to disk.

We have all experienced these "unexpected interruptions". Common types of these interruptions are kernel panics and crashes, power loss (brownouts and blackouts), turning your computer off without using the proper "Shut Down" procedure, and pressing the "RESET" button. Depending upon the exact moment of the interruption, you stand a good chance that your directory was either not updated correctly or some information was not saved to the disk. You now have directory damage or information loss.

When rebuilding, DiskWarrior scavenges the disk to find all the data necessary to build a replacement directory from scratch entirely in memory. This technique allows the user to preview the disk without modifying it and risking any data.

DiskWarrior's powerful scavenging engine searches the disk for the necessary directory data. In most cases, this data is good and is all that is needed to rebuild the directory. However, this file and folder information can also be damaged. DiskWarrior will determine the quality of any damaged data and, depending upon its quality, repair the damaged data and use it to rebuild the replacement directory. In our experience, no other utility matches DiskWarrior for its ability to scavenge and rebuild Mac disk directories

DiskWarrior fixes wrapper volumes, master directory block and alternate master directory blocks (HFS), volume headers and alternate volume headers (HFS Plus), volume bit maps, catalog trees and extents trees. There are also scores of items within these structures that DiskWarrior can repair.

But sometimes repairing the problem at hand is just not enough. Preventative maintenance is the key to having a healthy hard drive. As hard drives get older, the drive mechanisms wear out and begin to malfunction. Eventually, the malfunctions become so severe that the drive simply stops working. DiskWarrior can be configured to automatically check for possible drive malfunctions, further protecting against data loss. With DiskWarrior installed, users can enable DiskWarriors's hardware monitoring capability, which activates the S.M.A.R.T. (Self Monitoring Analysis And Reporting Technology) routines at regular intervals and checks the results.

Many times applications (even the Finder itself) just start to behave in weird ways — strange behavior, crashing, and more. We have found, repeatedly, that that the best thing to do is run DiskWarrior at that point. At least half the time, the problems will go away. Some people even run it periodically before symptoms occur. This is a great way to approach your disks health as well.

DiskWarrior 3 is the latest version of this product and is specifically tuned to for Mac OS X running faster than it ever did before. Given all the files that Mac OS X brings to your drive, it's critical to get version 3.

DiskWarrior is available on a CD that will start up in Mac OS X on any Mac with a G3 or better processor and 128 MB RAM (256 MB recommended). For customers who have older PowerPC based Macs that are not capable of running Mac OS X, the DiskWarrior CD will start up an earlier version of the Mac OS, allowing these customers to run DiskWarrior version 2.1, also included on the DiskWarrior 3 CD.

It's not a matter of *if* you should buy DiskWarrior, but rather if you are going to wait for a problem to occur and then be in dire need. We recommend getting it now so that you simply have it on hand.

Additional information about DiskWarrior can be found at http://www.alsoft.com/DiskWarrior/details.html.

David Breffitt is a network administrator at Xplain Corporation and *MacTech Magazine*. You can reach him at netadmin@xplain.com.

Peachpit

Essential books for the creative community

Start the New Year Right!

Achieve your new year's resolutions with help from these great guides covering everything from wireless networking to golf to building a business on eBay. It's all here!

>> **The Wireless Networking Starter Kit, Second Edition**
By Adam Engst and Glenn Fleishman
0-321-22468-X • $29.99 • 560 pages

>> **TechTV's Mod Mania with Yoshi: A Guide to Customizing Your Computer and Other Digital Devices**
By Joshua "Yoshi" DeHerrera
0-7357-1405-3 • $15.99 • 176 pages

>> **Macromedia Flash MX Professional 2004 for Server Geeks**
By Nate Weiss
0-7357-1382-0 • $45.00 • 624 pages

>> **TechTV's Guide to the Golf Revolution: How Technology is Driving the Game**
By Andy Brumer
0-7357-1406-1 • $19.99 • 176 pages

>> **Sell it on eBay: TechTV's Guide to Successful Online Auctions**
By Jim Heid and Toby Malina
0-321-22376-4 • $24.99 • 176 pages

>> **Mac OS X Panther Killer Tips**
By Scott Kelby
0-7357-1393-6 • $29.99 • 275 pages

Save up to 30%! Become a Peachpit Club Member today!

Enjoy 10% off all books every day at peachpit.com, earn an additional 10% discount as a Peachpit Club Member, and save 10% on top of that with this one-time coupon! Simply go to www.peachpit.com/mactech0104 and enter coupon code EM-J4AA-MTMF at checkout. It's that easy!

Peachpit Press

New Riders

ADVERTISER/ PRODUCT LIST

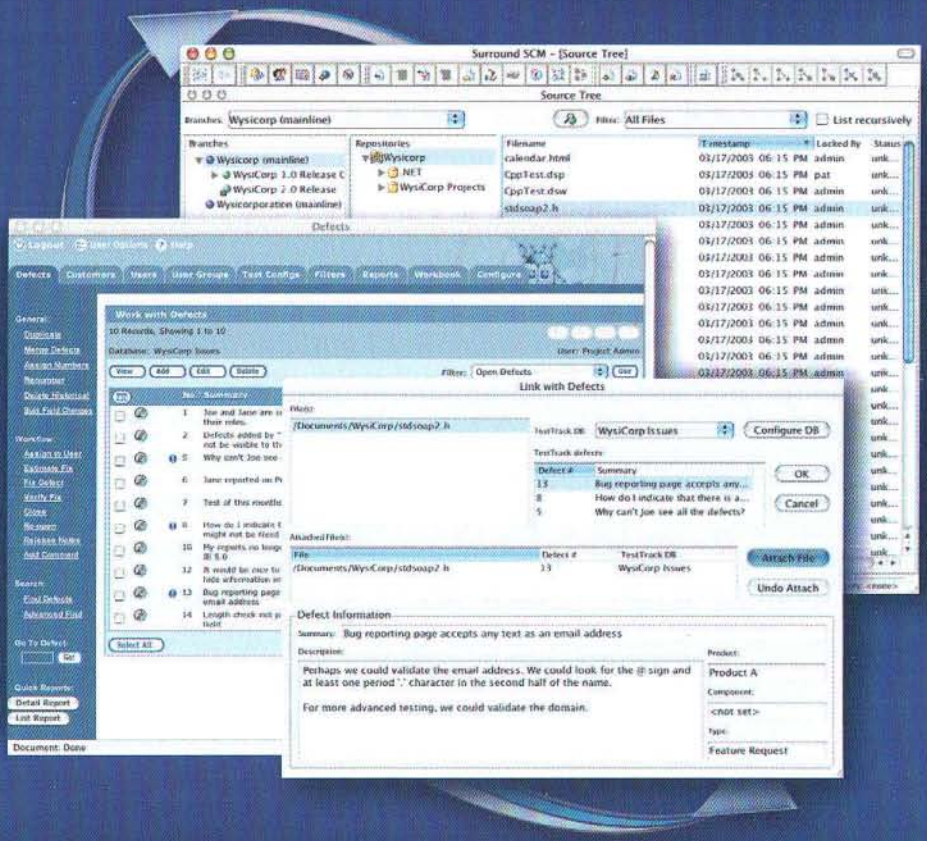
List of Advertisers

List of Products

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

Complete Source Control and Defect Management for Mac OS X

Seapine Software™
changing the world of software development



Effective source code control and defect tracking require powerful, flexible, and easy-to-use tools—Surround SCM and TestTrack Pro

- Complete source code control with private workspaces, automatic merging, role-based security, and more
- Comprehensive defect management — track bug reports and change requests, define workflow, customize fields
- Fast and secure remote access to your source files and defects — work from anywhere
- Advanced branching simplifies managing multiple versions of your products
- Link code changes with defects and change requests — know who changed what, when, and why
- Scalable and reliable cross-platform, client/server solutions support Mac OS X, Windows, Linux, and Solaris
- Exchange data using XML and ODBC, extend and automate with SOAP support
- Licenses priced to fit your budget

Seapine Software Product Lifecycle Management
Award winning, easy-to-use software development tools

Seapine **Surround SCM**

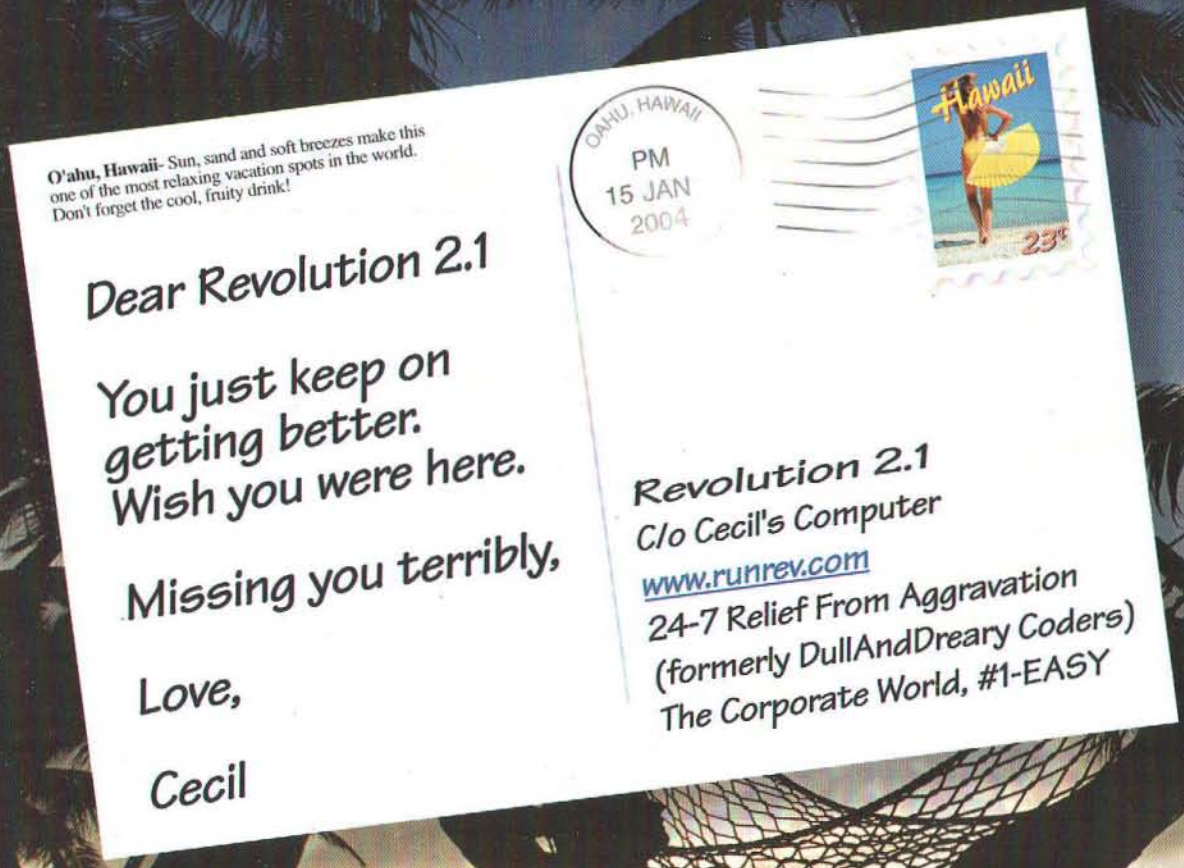
Seapine **TestTrack PRO**

software development 13th annual product excellence award

Download Surround SCM and TestTrack Pro at www.seapine.com or call 1-888-683-6456

all product names listed herein are registered trademarks of their respective owners. All rights reserved.

It'll Give You A Life.



But A Geek Is Always A Geek.

Revolution 2.1: the English like language designed around the way you think.

Develop and deliver on 14 platforms, including Mac OS X, Windows and Linux. Now with support for XML, additional SQL databases, video capture, Unicode, Reports, enhanced faceless CGI applications, and more.

And now from Dan Shafer, the first "how to" book on Revolution.
Get a headstart with "Revolution: Software at the Speed of Thought", volume one now shipping from www.runrev.com/revpress.
Thousands of developers have already joined the Revolution. Can you afford to wait?
Pricing starts at $149. Don't let the revolution in coding start without you.

Revolution Studio

winner macworld eddys

User Centric Development Tools

Come See Us At MacWorld San Francisco January 5th - 9th, 2004 Booth 935

RUNTIME REVOLUTION

Runtime Revolution • 91 Hanover Street • Edinburgh EH2 1DJ • UK
Phone +44 (0)131 718 4333 • Fax +44 (0) 845 4588487 • www.runrev.com • Email info@runrev.com